Universidade do Minho
Escola de Engenharia

José Pedro do Carmo Pontes

Velocity Control of a Two DOF Walking System

Outubro de 2012

José Pedro do Carmo Pontes **Velocity Control of a Two DOF Walking System**

UMinho | 2012

Universidade do Minho
Escola de Engenharia

José Pedro do Carmo Pontes

Velocity Control of a Two DOF Walking System

Dissertação de Mestrado
Mestrado Integrado em Engenharia Biomédica

Trabalho realizado sob a orientação da
Professora Doutora Cristina Manuela Peixoto dos Santos

Outubro de 2012

DECLARAÇÃO

Nome: José Pedro do Carmo Pontes

Endereço eletrónico: pontesjpc@gmail.com

Número do Bilhete de Identidade: 13352161

Título da Tese: Velocity Control of a Two DOF Walking System

Orientadora: Professora Doutora Cristina Manuela Peixoto dos Santos

Ano de conclusão: 2012

Designação do Mestrado: Mestrado Integrado em Engenharia Biomédica

Universidade do Minho, ___/___/_____

Assinatura: _____

Aknowledgements

I gratefully acknowledge the opportunity made by the Professor Cristina Manuela Peixoto Santos to integrate the ABSG group, and for introducing me to several very interesting subjects. I am grateful for the opportunity to learn as much as I could. I would like to thank all the help she gave me through the execution of my work, from the insightful discussions to the uncountable advice.

Lastly, I would like to acknowledge the good environment in the lab and the pleasant time spent with all my colleagues. Specially MSc. Vítor Santos for all the help he was patiently able to provide.

This work is funded by FEDER Funding supported by the Operational Program Competitive Factors - COMPETE and National Funding supported by the FCT - Portuguese Science Foundation through project PTDC/EEACRO/100655/2008. Pedro Pontes is supported by grant UMINHO/BI/78/2012.

Resumo

O controlo baseado em *Central Pattern Generators* (CPG) de sistemas locomoção permite um tipo de modulação que é direta e condicionada ao controlo dos parâmetros da rede. A implementação de tal controlo requer o conhecimento do mapeamento entre estes parâmetros e as variáveis que definem a locomoção. Uma abordagem a este problema consiste em usar métodos de *Machine Learning*. Neste trabalho, usando regressão linear múltipla, e, num estado mais tardio, *Locally Weighted Projection Regression* (LWPR), tentei manter a velocidade média de um sistema de locomoção de dois graus de liberdade. LWPR mostrou boas indicações de ser capaz de uma forma robusta e fácil de implementar de controlar certos aspectos da locomoção, havendo no entanto muito espaço para melhorar.

Abstract

The Central Pattern Generator (CPG) based control of locomotor systems allows for a type of modulation that is straightforward and constrained to the control of the network parameters. The implementation of such control requires the knowledge of the mapping between this parameters and the variables that define the locomotion. One approach to this problem consists in calling upon machine learning methods. In this work, using Multiple Linear Regression, and, in a later stage, Locally Weighted Projection Regression (LWPR), we try to maintain the average velocity of a two Degree Of Freedom walking system. LWPR showed good indications of being able to provide an easy to implement and robust way of controlling certain aspects of the locomotion, with there still being much room for improvement.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This manuscript presents the work I have been conducting for the past year, while part of the Adaptive System Behaviour Group at Universidade do Minho in Portugal. I took part in the group's research project, with biologically inspired robotic locomotion as a theme and dynamical systems as a tool.

The ultimate goal of the developed research is to improve and develop new controllers for articulated robots, create novel ways of achieving adaptive behaviors and general *know how* in the field of adaptive dynamic controllers.

1.1 Motivation

Humanoid robotics is often envisioned as one of the potential solutions to the service robotics problem and this field of research is exponentially growing. Scientific contributions in this domain go from intrinsically safe actuator design [40] to high-level social and cognitive interactions [6]. At the control level, several trends have been followed with various level of success.

The requirements for an autonomous robot to coexist with people and to be in highly dynamic and unpredictable environments are far too great and very demanding. Coordinating many degrees of freedom (DOF) in order to execute certain tasks is also a problem without a general solution. As such, developing solutions for achieving these requirements are still an important focus of study and research.

On uneven and rough terrains that may be comprised of several obstacles walking robots have clear advantage over conventional robots that use wheels or tracks. A walking robot contacts the ground in determined points, allowing the avoidance of obstacles. Furthermore, a walking robot can also be omnidirectional.

Despite walking robots providing such advantages when locomotion adaptation is the key, these robotic platforms are also hard to control. The controller of such a robot has to deal with a highly nonlinear system with many degrees of freedom, changes on body dynamics and unpredictable dynamics related to the robot-environment interaction. The robot must show adequate movements in order to support itself and propel itself through the environment, while not falling over.

There is great variety of approaches to design a controller for a walking robot. Some use pre-recorded trajectories to generate templates, others use stability criteria to do online trajectory modulation (locomotion control) [43]. The common point between most of them is the requirement of a perfect knowledge of the robot and environment dynamics.

The use of Central Pattern Generators (CPG) to implement control systems for real-time robot walking is a biologically inspired approach that has been growing in popularity in the past years. Robotic platforms have been based on animal structures for many years. Animals present innate abilities to adapt locomotor movements to changes in the environment, exhibit many corrective reflexes and are exceptional explorers of unstructured terrains.

These CPG models have been used to control many different type of robots with distinct types of locomotion [16]. Locomotor controllers have been implemented using several type of CPG models. Out of all these models, a system of coupled oscillators shows some interesting properties. The system has the ability to return to its rhythmic behavior after transient perturbations, and is also well suited for distributed implementation along the system to be controlled. By coupling the oscillators and the mechanical system, the natural dynamics of the combined system can be exploited in order to achieve locomotion [44].

These aforementioned type of CPG usually have few control parameters, which allow straightforward modulation of the locomotion. While the general end result achieved with the tuning of a parameter is usually easy to predict (e.g. lowering the parameter that controls the period of one stride should increase the locomotion speed), the exact relationship between these parameters and the locomotion they generate depends on the control architecture, the controlled system, external and internal perturbations, and the traversed environment, building up to a big number of variables to take into account, and making an analytical anal-

ysis both very hard and very costly to perform. Learning from experience is the a logical step over this difficulty, because it can provide an easier and more straightforward solution to the problem. Ideally, we want to obtain a mapping of the CPG parameters space and the locomotion predefined criteria. Changing the speed, direction, gait type and clearance, are just some of the possible final goals of the modulation.

1.2 Objectives

This work is a multidisciplinary undertaking that combines principles of dynamical systems theory, neuroscience and robotics. It will enable further contributions to the achievement of goal directed locomotion on an autonomous walking robot. Specifically, it will enable automatic adaptation to environmental changes for a locomotion system through the control of a small set of parameters.

The ultimate goal is to propose a learning control architecture, with a particular focus on adaptive goal-directed locomotion and, more specifically, in speed control.

In order to pursue this main goal it is necessary to achieve the following objectives.

1.

To design, a mathematical model for a Central Pattern Generator, taking in consideration features of its biological counterpart. The model will use nonlinear oscillators, which allow a fast response to stimuli. This makes it well suited for fast adaptive behavior because it

turns a high dimensional problem into simple selection of a small number of parameters that control the CPG network.

This model must enable modulation of the generated trajectories, possibly such that it reflects the environment changes. Nonlinear oscillators generate smooth trajectories modulated by simple parameters change [11].

The CPG network is intended to control a simulated walking system consisting in three segments and two rotational joints, suspended in a sliding support.

2.

In a first learning stage the goal is to obtain a relationship between some of the CPG system parameters and the mechanical system average speed throughout a set period of time. This allows for a first approach to velocity control of the designated system, and for a better understanding of the dynamics involved.

3.

On a second learning stage, the aim is to obtain a control structure that allows online learning, adaptation to condition changes, and a short training time, and apply it to the simulated locomotion system. The system is then to be tested in different sets of internal and external conditions, and afterward used that knowledge to maintain the average speed in a test run where those conditions were constantly changed. We apply a state of the art learning algorithm called Locally Weighted Projection Regression (LWPR) [42], that allows online learning, adaptation to condition changes, and a considerable decrease in training

time, which makes it a very desirable approach in order to achieve our objective.

1.3 Outline

This manuscript is organized as follows: Chapter 2 presents an non in-depth view of the nervous systems of vertebrate animals and its circuits involved on locomotion. Chapter 3 contains a brief review of the most revelant work done on the topics discussed on the next chapters. Chapter 4 describes the locomotion system simulated on the experimental part of this work, along with its controller and a more detailed explanation of the tackled control challenges. Chapters 5 and 6 expose the two different approaches used to ultimately attain the initial objectives, Multiple Linear Regression and Locally Weighted Regression, as well as the results obtained with each one. The last Chapter (7) summarizes and presents a discussion of the results, in addition to suggestion for future work.

1.4 Publications

The work carried throughout my participation in the group has led to an accepted submission as a conference participation.

José Pedro Pontes and Cristina P. Santos (2012), "Velocity Control of a Two DOF Walking System" Accepted for Numerical Analysis and Applied Mathematics ICNAAM 2012, AIP Conf. Proc. 1389; 19-25 September 2012, Kypriotis Hotels and Conference Center, Kos, Greece.

Chapter 2

State of the Art

For context and justification of the work exposed in this manuscript, this chapter will address, to some extent, the state of the art on Machine Learning applied to Robotics. The focus will be on methods applied to locomotion controllers based on nonlinear dynamical systems.

We will evaluate the advantages an disadvantages of the methods used in some of the most important papers in this area, taking into account the specific requirements of the objectives we wanted to achieve.

2.1 Learning control

Machine learning methods were introduced in automatic systems about four decades ago [39]. In the early 1980s there was an emergence of learning control with applications to dynamical systems, such as the development of learning control laws for mechanical systems and the discussion of their applicability on robot manipulators [2, 3]. Learning control refers to the process of acquiring a control strategy for a particular control system and a particular task.

The goal of learning control can generally be formalized in terms of finding a task-specific control policy [9]:

$$u = f(s,t,\alpha), \tag{2.1}$$

that maps the continuous state vector x of a control system and its environment, possibly in a time t dependent way, to a continuous control vector u. The parameter vector α denotes the problem-specific adjustable parameters in the policy π. As an example, given the current state of a robot we control, and the environment it traverses, we want to determine which control input (e.g. torque commands for joint actuators) we need to achieve a desirable outcome (e.g. a certain speed or path).

2.2 Model based learning

One simple approach is to learning control is to use methods of function approximation to estimate a forward model $f()$ that uses states and actions to predict outcomes (z) [4]:

$$z = \hat{f}(s,t,u), \tag{2.2}$$

where $\hat{f}()$ is an approximation of $f()$. Then a controller is computed based on the estimated model, which is a technique belonging to the category of model based learning.

Kawamura and Fukao [20] developed a method that interpolates input torque patterns obtained through Learning Control in order to create a desired motion with a different speed pattern or time-scale. Learning Control optimizes input patterns at each iteration by velocity or acceleration error (the different between a desired motion and the actual motion). Applying the algorithm in an actual robot can be time and memory space consuming, if we need many specified motion patterns. Interpolating ideal feedforward input patterns, generating another desired motion, helps in overcoming this difficulty. The authors demonstrated that they could form an arbitrary speed pattern from four motion patterns with different time-scales, if all the spatial trajectories were the same. This approach lacks automatic, online, adaptation to changes in the desired trajectory, not to mention the fact that it constrains the spatial trajectory used.

Model based learning results in indirect control, because it usually requires the computation of the controller after we estimate the model. If instead we aim to learn the policy directly, without detour through model identification, we will be using direct control, with model-free

learning.

2.3 Model-free learning

Model-free methods don't use any explicit information about the dynamics of the robot, the environment, and the interactions between both. This makes them, at a first glance, more appropriate for a task that demands adaptability of the control policy to changes in these systems.

Dynamical movement primitives are non-linear differential equations with attractor dynamics [17]. Their output serves as desired trajectories for a robot, which can be learned rapidly, and easily re-scaled in terms of the patterns' amplitude, frequency, and offset. They can be used, as an example, to constitute a dynamic systems model which represents a library of different movements [34]. Being non-linear differential equations, these primitives can also be used as a CPG of a robot. Nakanishi and colleagues [27] developed a framework for learning biped locomotion based on this idea, having demonstrated trajectories learned through movement primitives by locally weighted regression. This method provides flexibility in encoding complex movements and the potential capability of improving learned movements through reinforcement learning, with the downside of requiring the demonstration of the learned trajectories.

Gams and others [11] produced another work that fits the view that biological movements are constructed of motor primitives: a dynamical system that learns and encodes a periodic signal. The system has two layers, one that uses nonlinear oscillators to extract the

fundamental frequency of the input signal, and a second that uses nonparametric regression techniques to learn the waveform, by shaping the attractor landscapes according to demonstrated trajectories. The system requires no prior knowledge of the frequency and waveform of the input, and is able to modulate the learned trajectory in response to external events. The adaptation for a signal with six frequency components achieved identical input and output signals. The system can be expanded to several dimensions, working in parallel for various Degrees of Freedom (DOF) of the signal. The authors tested this approach with a humanoid HOPA-2 robot, using 8 DOF to control de arms. They began by using the system to learn and reproduce simple 2D trajectories made by the end effector, and then tested for more complex trajectories. The system successfully learned both patterns, with frequency adaptation taking longer on the complex patterns due to multiple frequency components.

2.4 Reinforcement learning

Reinforcement learning (RL) studies how systems can learn to optimize their behavior in order to obtain rewards and avoid punishments. As an example, an RL based controller for a biped robot is analogous to a baby's acquisition of biped locomotion along its growth [25]. Large applications of this type of learning require the use of function approximators. Historically, the first approach for this approximation has been estimating a value function, with the action-selection policy represented implicitly as the policy that selects in each state the action with highest estimated value [37, 38] - this is model-based control. In robot control, the computation of the value function or its approximation is difficult, analytically or numerically, because of the enormous size of the state and action spaces. Policy gradient reinforcement learning methods avoid this issue by approximating a policy directly using a

function approximator independent of the value function - a model-free solution. The policy is updated according to the gradient of expected reward with respect to policy parameters [37].

Endo and colleagues [10] developed a learning framework for a CPG-based biped locomotion controller using the aforementioned method. They acquired an appropriate feedback controller within a few thousand trials and the controller obtained in numerical simulation achieved stable walking with a physical 3D hardware full-body humanoid in the real world. Mori and others [25] used a CPG-actor-critic model - a policy gradient method incorporating a value function - which approximates a lower-dimensional projection of a value function, instead of a true value function, which is often an easier and lighter approach. In this method, the actor is a controller that transforms an observation into a control signal, and the critic approximates the value function to predict the gradient of the average cost toward the future. The actor's parameter is updated so that the cost predicted by the critic becomes small. The RL method was applied to autonomous acquisition of biped locomotion by a biped robot simulator. Computer simulations showed the method was able to train a CPG controller such that the learning process was stable. In order to escape local optima, the actor parameter was re-trained by reinitializing the critic's parameter, which resulted in a lot of training episodes. This makes it difficult to apply the method directly to real robots.

One possible solution to such problem is to use an approach that does not require stoping or reseting. Sproewitz and colleagues [36] presented an approach that also uses CPG, but with a gradient-free optimization algorithm - Powell's method. The method achieves online learning by running the optimization algorithm in parallel with the CPG model, with speed of locomotion being the criterion to be optimized. Powell's method is fast, but presents more risk of converging to a local optimum than stochastic methods (such as genetic al-

gorithms, particle swarm optimization or simulated annealing). Speed result obtained with two gaits from experiments with a quadruped robot showed that the speed can be adjusted monotonically with the frequency and, interestingly, the relation is almost linear in the given frequency range.

Matsubara and others [23] combined a model-based Center of Mass (COM) controller and a model-free RL method to acquire dynamic whole-body movements in humanoid robots. A purely model-based controller considers highly approximated dynamics, which can cause poor tracking performance, and it is affected by modeling errors. While the model-based controller can cope with high-dimensionality, the RL method can improve its performance. The COM controller derives joint angular velocities from the desired COM velocity, while RL is used to acquire a controller that derives the desired COM velocity based on the current state. The authors set the goal of strengthening ball-punching on a Hoap-2 humanoid robot in numerical simulation, through a learning process that focused on a COM movement. The locally optimal punching motion with maximal reward was acquired around 2000 episodes of learning and the acquired cooperative whole-body movement, was shown to be effective even in a real environment.

2.5 Learning in CPG based control

Modeling with nonlinear dynamics systems is mathematically quite difficult; optimization approaches are often much easier to handle with well-established algorithms and software tools [34]. This approaches also have the advantage of giving the opportunity to provide a more general, and more compact, representation of a desired trajectory [18]. Trajectory

modulation allows for the on-line adaptation to changes in goals and the system's environment [11].

Manually tuning the open parameters of a CPG network, in order to achieve a desired behavior, is a cumbersome, lengthly and inexact process [27]. The speed of locomotion, for example, cannot be expressed as an analytical function of the robot parameters because it depends on the dynamics between the robot and the environment [36]. An optimization algorithm can run in parallel with a CPG network and update its parameters. With differential equations of second order, even abrupt parameter changes will result in smooth convergence towards a new limit cycle, after a short transition period, which means the robot doesn't need to be stopped between iterations [11, 36].

Evolutionary methods are often used of find the parameters of CPG networks. Saif [32] applied the standard Genetic Algorithm (GA) and the Empire Establishment Algorithm (EEA) - a parallel GA - to approximate such parameters, but found that with that approach learning to walk was very time-consuming. He then made and adaptive oscillator that can rapidly learn arbitrary periodic signals in a supervised learning framework, and is completely embedded in a dynamical system. He used a 3D simulation of a NAO biped robot, training the adaptive oscillator with sample trajectories provided by a player of a ROBOCUP competition. The NAO robot showed a stable and fast walking pattern after 1.5 hours of adaptation.

Sato, Watanabe and Igarashi [33] proposed a combination of GA and Reinforcement Learning for determining parameters of a CPG network, and applied it to a quadruped robot. The CPG inner coefficients are learned by a GA, and the feedback controller and the connection controller are learned by Reinforcement Learning. Additional learning after acquiring a walking behavior is done only by RL, with the inner parameters defining a basic walking rhythm.

They confirmed that the robot could adapt to new environments by only learning sensory feedbacks and connections among oscillators. Christensen, Larsen and Stoy [8] adopted applied a stochastic optimization learning algorithm to optimize eight open parameters of a CPG network used to control a gait. They used the model-less Simultaneous Perturbation Stochastic Approximation method, which requires only two robot trials with different controllers per iteration, independently. They applied the algorithm in an online gait learning experiment on a quadruped robot. The robot, while learning, improved its initial velocity from $0\ cm.s^{-1}$ to $13\ cm.s^{-1}$ after eight minutes, and with the final gait learned, without online learning, it moved at $17.5\ cm.s^{-1}$, which is faster than a manually designed gait. The strategy was successfully applied in two systems with different degrees of freedom and modularity.

2.6 Locally Weighted Learning

Locally weighted learning uses locally weighted training to average, interpolate between, or extrapolate from training data. The most sophisticated approaches to this type of learning typically present a learning structure that allows a multitude of models to join together to make a global approximation, by assigning to each one a certain weight. Each local model is trained independently such that its total number does not affect how complex a function can be learned - complexity is only controlled by the level of adaptability of each model [5].

RFWR is a learning algorithm based on locally weighted learning that uses nonlinear function approximation with structure adaptation, represented by piecewise linear models. Nakanishi, Ferreal and Schaal [26] compared four learning schemes for a simple function approx-

imation: Locally Weighted Regression (LWR), LWR with normalized weights, Radial Basis Functions (RBF), and RBF with normalized weights. They found that the RBF networks need more re-organization of previously learned parameters when they increased the number of the local models for the approximation, and that they strong cooperation may cause negative interference on the approximator domain. In addition, both RBF approaches started overfitting at a certain number of local models, while the LWR methods approximation improved.

An automatic structure adaptation of the function approximator is useful when the complexity and structure of the the function to be approximated are not know beforehand [26].

Chapter 3

Bio-Inspired Architecture

It is clear that animals surpass current robots on walking and moving around in our natural world. On their movements they exhibit many corrective reflexes when faced with unexpected perturbations, and present an exceptional adaptability in rough terrain.

Throughout many years of evolution the locomotor circuits in the nervous system were extended and improved. These circuits rise in complexity from the small fish to the walking mammal, but share similarities in organization and function which were conserved through evolution.

We take inspiration from nervous systems in hope that these potential mechanism of animal motor control can help on improving the design of adaptive algorithms and controllers, while never abandoning an engineering perspective.

3.1 Neural structures for locomotion in vertebrates

The nervous system is a network of specialized cells that control all bodily functions. It is responsible for sending, receiving and processing nerve impulses throughout the body, controlling all the organs and muscles. As can be seen in Figure 3.1, the nervous system in vertebrate animals is divided in two main parts: the peripheral nervous system (PNS) and the central nervous system (CNS) [19].

The PNS consists in nerve cords constituted by afferent fibers that relay sensory information from the limbs and organs to the CNS, and by efferent fibers which transmit information from the CNS to organs and limbs [14].

The spinal cord receives and processes peripheral sensory information from the skin, muscles and limbs, and relays it to the brain. It is divided, from head to trunk, into cervical, thoracic, lumbar and sacral regions. It contains neural circuits that endogenously generate rhythmic patterns. There are several of these circuits in the spinal cord, controlling the rhythmic activity for breathing, swallowing, chewing and walking [19, 14].

3.2 Central Pattern Generators

Central Pattern Generators (CPGs) are found in all vertebrate animals, including humans.They are intrinsic spinal networks composed by rhythmogenic units that carry the endogenously generated limb muscle activation during locomotion [12, 22, 21]. The CPGs are activated through tonic signals from supraspinal regions (Figure 3.2).

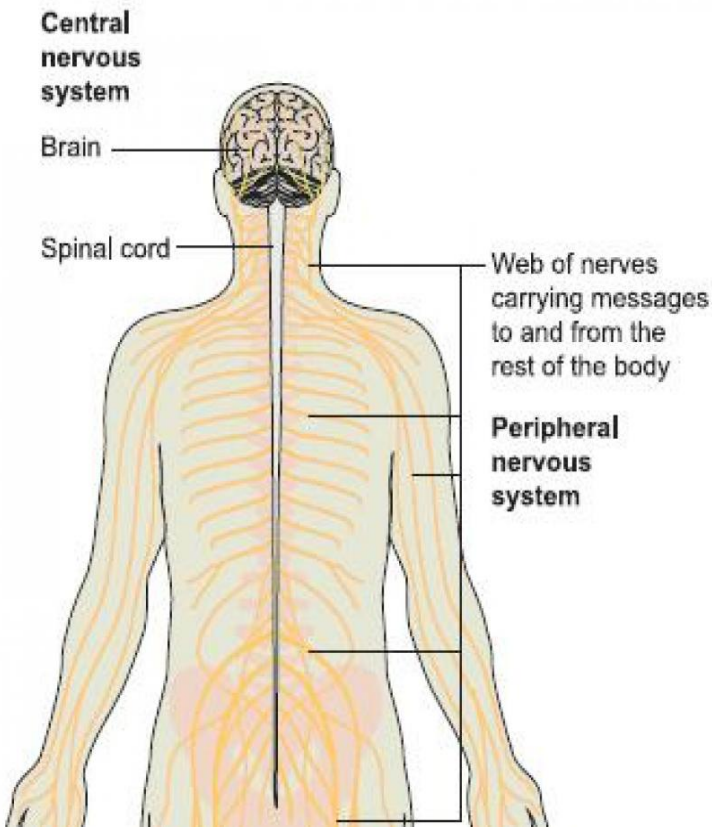


Figure 3.1: Basic concept of the central nervous system and the peripheral nervous system on a vertebrate [1].

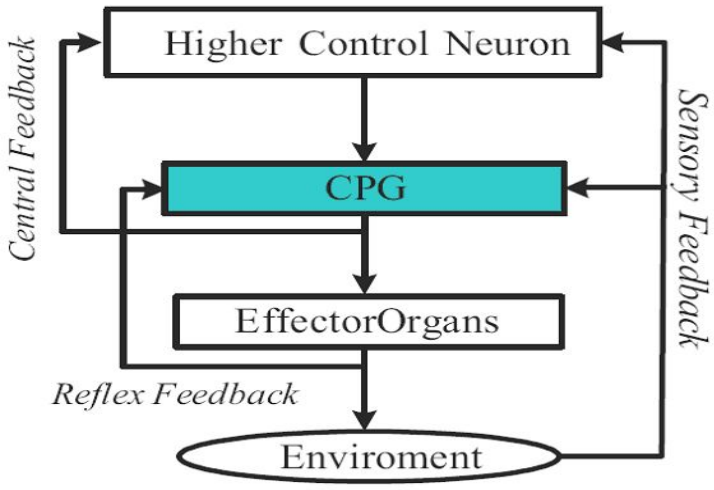


Figure 3.2: Basic concept of the central nervous system and the peripheral nervous system on a vertebrate [1].

These circuits were extensively studied: in fish, as the lamprey; in amphibians, like tadpoles, frogs, toads and newts; in reptiles and birds; and in mammals, as cats and dogs [12].

Several experiments in animals show that after transection of the spinal cord and after afferent input is abolished, rhythmic locomotor movements are exhibited when applying certain excitatory signals. Also, it was evidenced the generation of fictive locomotion in several spinal preparations. These studies, along many other experiments have provided detailed information about the CPGs and the effects of the sensory information on its generated patterns.

It has been proposed that the CPG for each limb is composed by smaller rhythmogenic circuits, the unit-CPG, each controlling one muscle group of extensors and flexors of a limb, i.e. one unit-CPG controlling one joint in a limb [13]. The organization of the CPG is very

important when considering the required flexibility when generating the different varieties of limb movements during goal directed locomotion [14]. This intralimb coordination of the generated pattern depends of the limb movements to perform - when walking in different directions the unit-CPGs must be coordinated in different ways in order to generate a different activation pattern.

The CPG provides the basic rhythm output for locomotion while integrating powerful commands from various sources that serve to initiate or modulate it, meeting the requirements of the environment. They show adaptation to different gait patterns and different walking contexts. Signals from supraspinal, spinal and peripheral structures are continuously integrated by the CPG for the proper expression and short-term adaptation of locomotion, providing a great versatility and flexibility on the performed movements [31, 28].

Chapter 4

Velocity control of a two DOF walking system

4.1 System description

The system used for the simulation (figure 5.1) is comprised by a structure with the very basic structure of a leg intended for hopping, and a supporting sliding platform. The "leg", given this setup, produces a motion similar to a hopping motion, and shall be from now on referred to as the hopper. The hopper has three segments and two actuated joints. The two lower segments are akin to the thigh and foot of a vertebrate's leg, with the joints being comparable to the hip and the ankle. The upper segment keeps the hopper connected to the sliding platform.

With movement from the hopper, the sliding platform will move forward or backwards along the tracks that supports it. This helps the hopper's balance, as well as restraining the movement along a fixed axis.

The work was implemented on this system because of the facility to implement in it a central pattern generator inspired control, and because of the stability it normally offers. As the goal is to analyze the ability to finely tune the parameters of the network to achieve a desired speed, we considered aspects such as the robot's stability and direction control to be secondary, as well as, at least initially, the complexity of the system. The reasoning being that the applied algorithm should be easily expanded to higher dimensions, and capable of processing the extra quantity of information.

The system was simulated on a robot simulation environment called WEBOTS [24], which was developed as a research tool for investigating various control algorithms in mobile robotics.

4.2 Locomotion controller design

The CPG was implemented through the use of a nonlinear dynamical oscillator that presents a Hopf bifurcation [30]. The oscillator is presented as follows,

$$\dot{x} = \alpha(\mu - r^2)(x - O) - \omega y \tag{4.1}$$

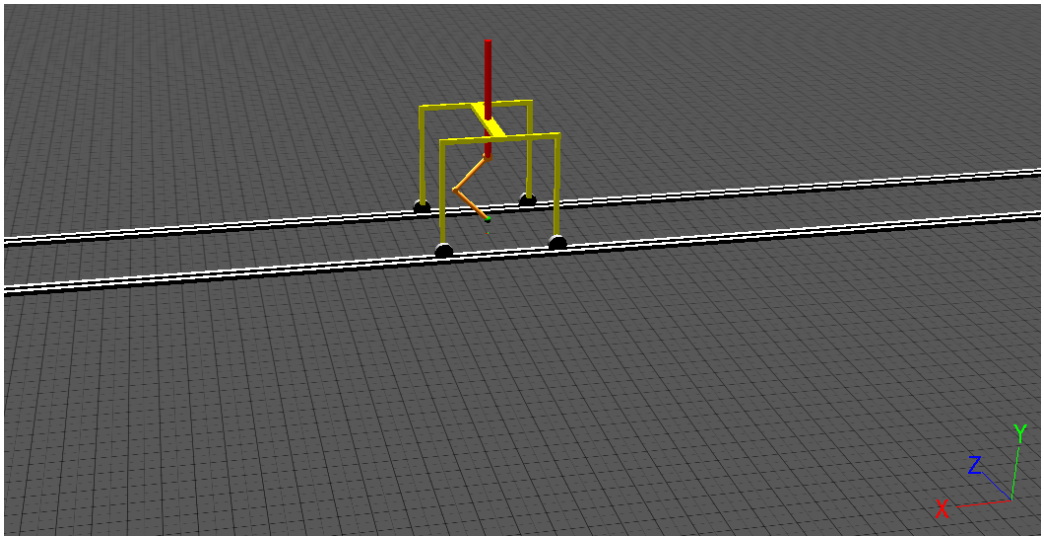


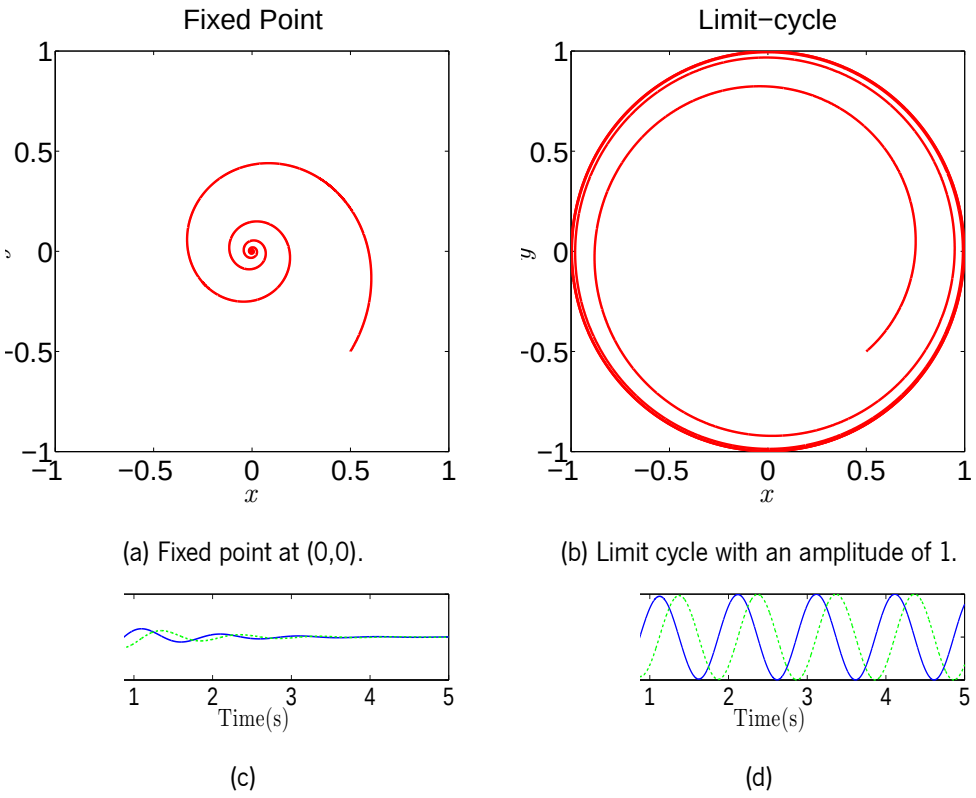
Figure 4.1: Simulated environment for the second stage testing.

$$\dot{y} = \alpha(\mu - r^2)y + \omega(x - O) \tag{4.2}$$

where x and y are the state variables that present oscillatory harmonic solutions or a stable fixed point, and $r = \sqrt{(x-O)^2 + y^2}$. The variable ω represents the frequency of the oscillator, $\sqrt{\mu}$ represents the amplitude of the oscillations, and α is a positive constant that controls the speed of convergence to the limit cycle. The variable O is used to control the offset for the solution in the x state variable.

This oscillator relaxes to $x = O, y = 0$ for values of $\mu < 0$, and follows a stable orbit when $\mu > 0$ (Figure 4.2).

Using different speeds of convergence for x and y allows to have a faster convergence on the y axis while having a limited derivative for a control policy generated by x. In Figure 4.3 we



(a) Fixed point at (0,0).

(b) Limit cycle with an amplitude of 1.

(c)

(d)

Figure 4.2: Solutions for the oscillator when (a), (c) $\mu = -1$ and (b), (d) $\mu = 1$. Initial condition $(x_0, y_0) = (0.5, -0.5)$, $O = 0$, $\alpha = \beta = 1$, $\omega = 6.3 rad.s^{-1}$. In (c) and (d) the x state variable is the solid blue line and the y state variable is the dashed green line.

can see that the oscillator solution with the larger value for the parameter β, which controls the speed of convergence of the y state variable, shows a faster convergence towards the y axis.

The parameter ω specifies the frequency of the oscillations in $rad.s^{-1}$. The period of the oscillations, or duration of a cycle, T is given by $T = \frac{2\pi}{\omega}$, in seconds. Changing the signal of ω changes the direction of the limit-cycle: for $\omega > 0$ the limit-cycle rotates counter-clockwise, and for $\omega < 0$ it rotates clock-wise. In Figure 4.4 we can see the effect of both changing the signal and the module of ω. Notice that when the parameter is altered the oscillator promptly changes the frequency of the generated solution, resulting in a smooth response. We can also verify that with the change in the direction the state variable that was previously trailing behind in time will get ahead (y was trailing behind initially, and then gets ahead after the reverse in direction at the 2 seconds mark).

The generated trajectories of this oscillator can be summarized as

$$\begin{bmatrix} x(t) \\ y(t) \end{bmatrix} = \begin{cases} \begin{bmatrix} O \\ 0 \end{bmatrix} & ,\mu < 0 \\ \begin{bmatrix} \sqrt{\mu}\cos\omega t + O \\ \sqrt{\mu}\sin\omega t \end{bmatrix} & ,\mu > 0 \end{cases} \tag{4.3}$$

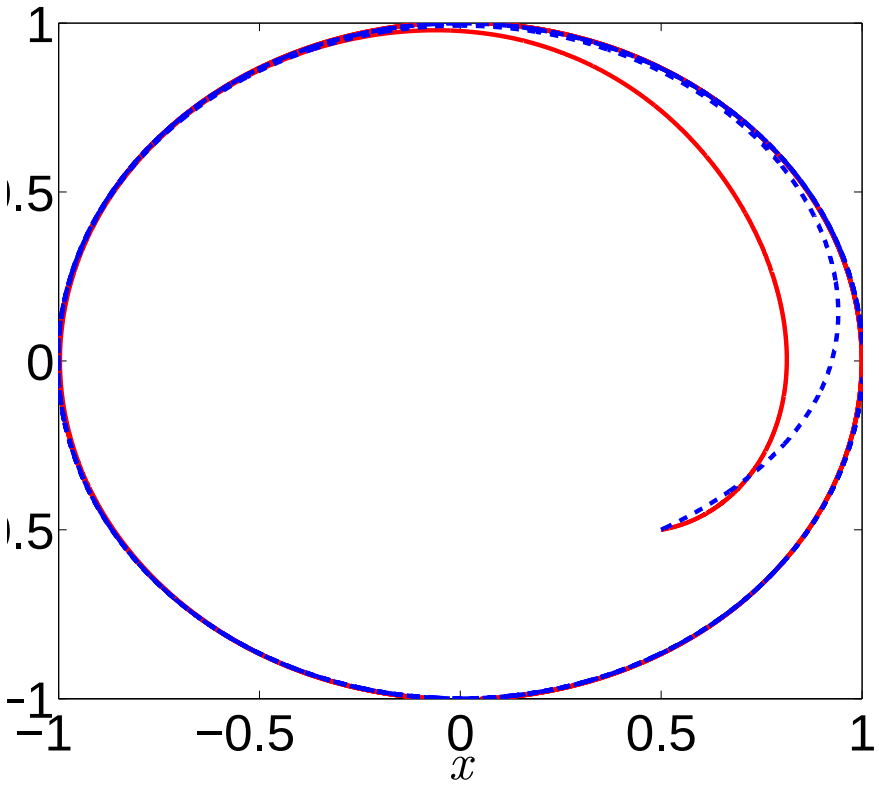


Figure 4.3: Two different solutions for the oscillator with $\mu = 1$, $O = 0$, $\omega = 6.3 rad.s^{-1}$ for both solutions. The solid red line represents a solution with $\beta = 10$ and $\alpha = 0.1$ and the dashed blue line represents a solution with $\beta = 0.1$ and $\alpha = 10$.

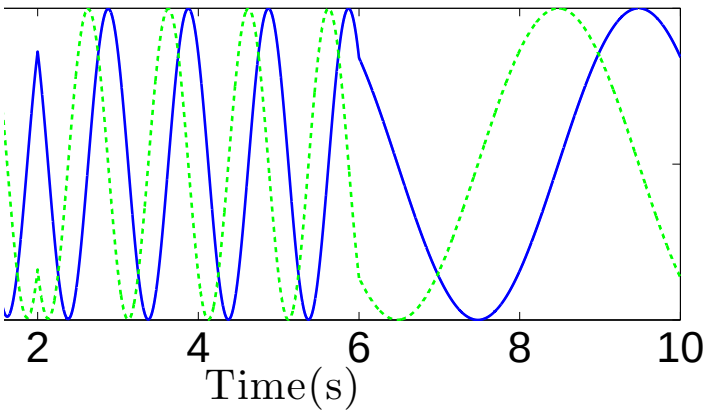


Figure 4.4: State variables of the Hopf oscillator, in function of time, for a specific solution with $\mu = 1$, $O = 0$, and $\alpha = \beta = 1$. The solid blue line represents x and the dashed green line y. The parameter ω starts at $6.3 rad.s^{-1}$, is changed to $-6.3 rad.s^{-1}$ at $t = 2s$ and to $-1.57 rad.s^{-1}$ at $t = 6s$.

4.2.1 Step phases frequency modulation

For $\omega > 0$, the oscillator will be performing the swing phase of the locomotion when $y < 0$ (with x in a ascending phase), and the stance phase when $y > 0$ (with x in a descending phase). The relation between the length of these two phases is the duty factor λ, given by

$$\lambda = \frac{T_{stance}}{T_{stance} + T_{swing}} \tag{4.4}$$

As it is defined, the oscillator generates trajectories with both phases in equal duration, which gives $\lambda = 0.5$. To specify the overall step cycle, ω need to be changed, which will also change the duration of both step phases. This is an aggravating constrain, since there are advantages in having step phases with different duration, such as changing the speed

of locomotion, and obtain a different gait [12, 15]. Righetti [30] employed a mechanism that modulates the value of the frequency in each phase of the step

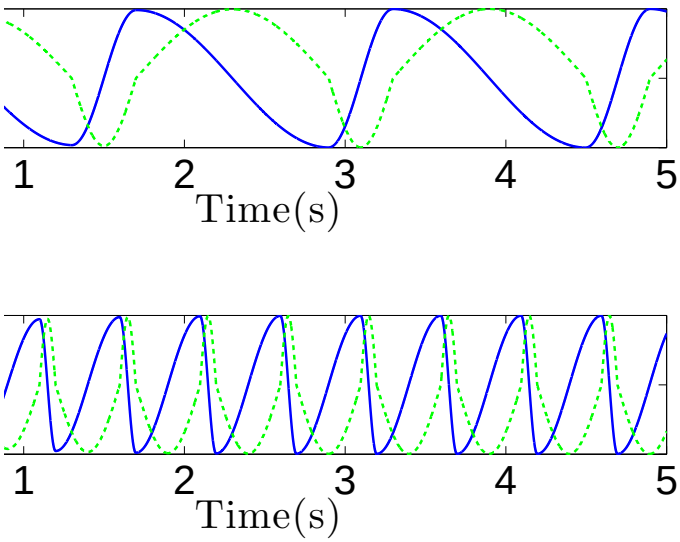
$$\omega = \frac{\omega_{stance}}{e^{-by}+1} + \frac{\omega_{swing}}{e^{by}+1} \tag{4.5}$$

This equation alternates between two values for the frequency ω, depending on the step phase identified by the value of the state variable y. The frequency of oscillation during these two phases is specified by the value of $\omega_{swing} = \frac{\pi}{T_{swing}}$ and $\omega_{stance} = \frac{\pi}{T_{stance}}$. The value b controls the alternation speed.

4.2.2 The CPG network

The basic outputs of the CPG state variables x and y are sinusoidal waves, which is stated in equation 4.3 and can be observed in Figure 4.2(d). The variable x was defined as the one that encodes the value of a joint's angle. The sinusoidal waves of the output of the x variable of the oscillators are translated into position commands for the actuated joints, resulting in a smooth locomotion. This translation is done internally by Webots, where the user specifies a desired position, and then a P-controller takes into account the desired position, computes the current velocity of the joint, and then calculates the necessary velocity to achieve that position.

The controlled system is composed of two joints, which means we will need a network of two simulated CPGs in order to have the desired control. If we apply a control system based on a network of CPG to a system with more than one limb, we will have to coordinate the



(a) Limit cycle with an amplitude of 1.

Figure 4.5: Solutions for the oscillator with $\beta = 0.75$ (top) and $\beta = 0.2$ (bottom), for a $\omega_{swing} \approx 7.85 rad.s^{-1}$ ($T_s w = 0.4s$). The solid blue line represents x and the dashed green line represents y. The duration of the swing phase — negative phase of y (the ascending phase of x) is kept constant — and only the the stance phase duration changes.

phases of the different units in order to achieve a certain gait — this is called interlimb coordination [29].

In this case, the coordination of the CPG units is done in two ways. In first place, the frequency parameter of the oscillator that controls the "ankle" is set to the double of the frequency parameter of the oscillator that controls the "hip". Then, we have to modulate the output of the ankle, since there is a need for a slight flexion during the swing phase, in order to lower the height of the systems' center of gravity, providing a better support for its body. This was achieved by modulating the parameters O_k and μ_k according to whether the locomotion is on the stance or swing phase (the subscripts "k" and "h" refer to the CPG parameters and variables used in the ankle and hip oscillator, respectively.). This modulation is the same one presented in (4.5).

$$O_k = \frac{O_{k,st}}{e^{-by_k}+1} + \frac{O_{k,sw}}{e^{by_k}+1} \tag{4.6}$$

$$\mu_k = \frac{\mu_{k,st}}{e^{-by_k}+1} + \frac{\mu_{k,sw}}{e^{by_k}+1} \tag{4.7}$$

4.3 Problem formulation

The shortest term objective consists in finding the appropriate CPG parameters that result in a locomotion that exhibits certain characteristics. Specifically, the goal is to modify the parameters in order to reach a certain speed, but one of the main objectives also lies in finding a more global approach, so that other characteristics of the locomotion (e.g. the

COM trajectory) can be optimized with easy modifications to the approach. This problem can be tackled as a learning control problem, in which we need to obtain a model that represents the task and the environment at hand, train this model, and decide how it should be used to control our system.

Our model will try to establish the relationship between a number of variables, which can be separated in three groups. The state vector s contains the variables that we can observe, but not choose. The system's center of mass (COM) or ground reaction force (GRF) are examples of variables that can belong to the state vector, in this specific case. The action vector u groups the variables that we can control directly, and ultimately pretend to be automatically chosen with the aid of our model, such as the CPG amplitude and gait period parameters. Finally, the output vector z will contain the locomotion defining quantities, that we wish to be able to control through a change of the action vector. The state and action vectors should be chosen according to the output vector [5].

4.3.1 Control using a forward model

In the first stage of the work, we used a forward model to control the parameters. A forward model uses states and actions to predict outcomes:

$$z = \hat{f}(s, u), \tag{4.8}$$

where $\hat{f}()$ is a approximation of $f()$, which is not known at the beginning of the task.

This approach requires three steps. First, we need to collect training data for the model. This can be achieved by simulating the system in a variety of scenarios that present a variety of values for the variables chosen to form the action vector. If done correctly, this will provide enough information to approximate the forward model. The amount of time and computational cost put in the training, as well as the values chosen for the different variables depends on the accuracy desired for the control of the system on a specific range of values of the output, i. e., if we want to control the system for velocities between $0.1m.s^{-1}$ and $0.5m.s^{-1}$, we need to find the required values of the parameters of the network that results in values between, and around that range.

The second step is the approximation itself. Here, two decisions have to be made: what are we going to model the approximated function to, and what tool are we going to use do that approximation [35]. Choosing to use a specific, analytically well defined target function, is the easiest and most direct choice, but brings two problems. In the first place, choosing a function with limited knowledge of the relationship between the variables we want to obtain, which is the case, can prove to be a complete shot in the dark. If the function can't be modeled with a linear relationship, we need to decide on the correct term for each variable. The second problem is the fact that the function may be impossible to model with a specific, well defined function, in a context practical way. The response of the locomotion system in regards to the controlled parameters may change with time, which may render the previously approximated model less accurate, or even useless. This being said, we decided on such an approach when the problem was first tackled. A simple, and easy to approximate polynomial model is flexible enough in its terms to provide valuable information to better understand the problem faced and attain a more practical and correct approach to the control problem.

The third step is to decide the way the control is ultimately done, i.e., the way actions are chosen to achieve the pretended output. To use a forward model for control, we need more than a single lookup. A numerical inversion of the model is needed to search for an action vector that is predicted to achieve the outcome. This is identical to numerical root finding over an empirical model, and we can apply the same approaches in both processes [5]. With a polynomial model, we can apply Newton's method with excepted success, providing a good initial approximation, as the function's derivative is easily obtained.

The initial approach using Multiple Linear Regression for the approximation and Newton's method to aid in the control policy is presented in chapter 5.

4.3.2 Control using an inverse model

In the second staged, we used an inverse model, which uses states and outcomes to predict the necessary action,

$$u = \hat{f}^{-1}(s, z). \tag{4.9}$$

An inverse model usually provides an easier way to chose an action to provide a desired outcome, as the action vector already is the output of the function, but it requires more sophisticated approaches to do the approximation. The choice of the model is a particularly sensitive one. Having an easier control policy, we should aim for a model that better suits this challenge. The algorithm Locally Weighted Projection Regression (LWPR) [41] achieves nonlinear function approximation, using various local linear models to model the function.

Each local model has a weight that is used to find its contribution to a specific output. This change in the approach to the way the function is modeled provides the ability to approximate a relationship that is not known beforehand, and that changes with time.

This approach using LWPR for the approximation, and a more direct control policy is presented in chapter 6.

Chapter 5

Velocity control using a forward model

The purpose of this stage of the work was to get a better understanding of the velocity range limitations of the hopper, how that velocity behaves with different parameters and environment conditions, and what is the better approach to successfully control the system for a desired velocity.

5.1 Velocity characterization conditions

Nine experiments were conducted on a first stage, were the goal was to evaluate how the change in two of the network parameters (the hip oscillator's amplitude, μ, and the

hip oscillator's swing phase duration, T_{swing}) would affect the mean velocity of the leg in different environments and mass values for the leg 's lift. The environments used were a floor without inclination or irregularities, a ramp, and a floor with bumps. The different mass values used were 1Kg, 5Kg and 10 Kg. The conditions used in each of the nine experiments, for the first stage, were as follows:

1. Normal floor, 1Kg lift;
2. Normal floor, 5Kg lift;
3. Normal floor, 10Kg lift;
4. Bumps, 1Kg lift;
5. Bumps, 5Kg lift;
6. Bumps, 10Kg lift;
7. Ramp, 1Kg lift;
8. Ramp, 5Kg lift;
9. Ramp, 10Kg lift.

In a second stage, the parameter T_{swing} was set to a constant value, and the objective was to maintain the hopper 's mean velocity on a single run, where the leg trespassed the different environments used on the first stage, and its lift mass values were also changed accordingly.

5.2 Velocity characterization

In all of the nine experiments, the system 's locomotion was controlled (through position control) for 25 seconds for each set of values for the evaluated parameters. These parameters where the hip oscillator's amplitude, μ, and the hip oscillator's swing phase duration, T_{swing}. The parameter μ was evaluated for values starting at 5, ending with 40, and a step interval of 1. The swing phase duration was evaluated for values starting at 0.3 seconds, ending at 1 second, and a step interval of 0.02 seconds.

The first, second and third experiments were conducted on the WEBOTS world shown in figure 5.1. In the fourth, fifth and sixth experiments, the floor was changed in order to contain some bumps, as shown in figure 5.2, and in the seventh, eighth and ninth experiments a ramp was added to the floor (figure 5.3).

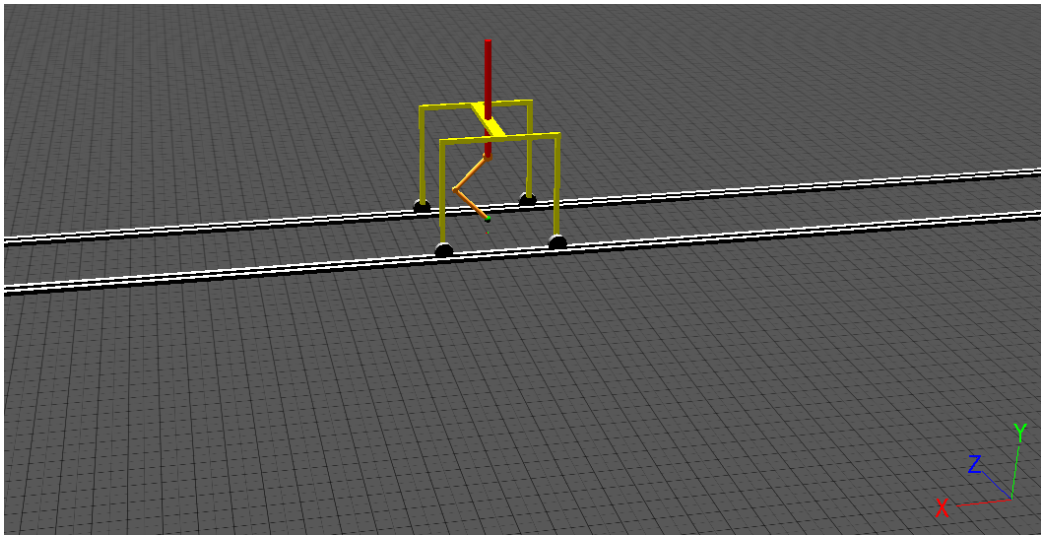


Figure 5.1: Simulated world on WEBOTS, for the first, second, and third experiments of the first stage.

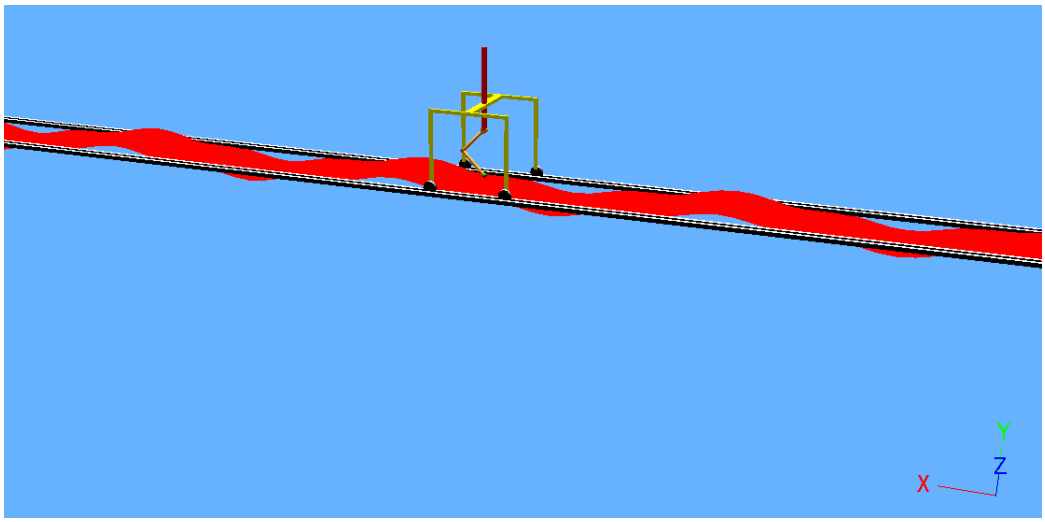


Figure 5.2: Simulated world on WEBOTS, for the fifth, sixth and seventh experiments of the first stage.

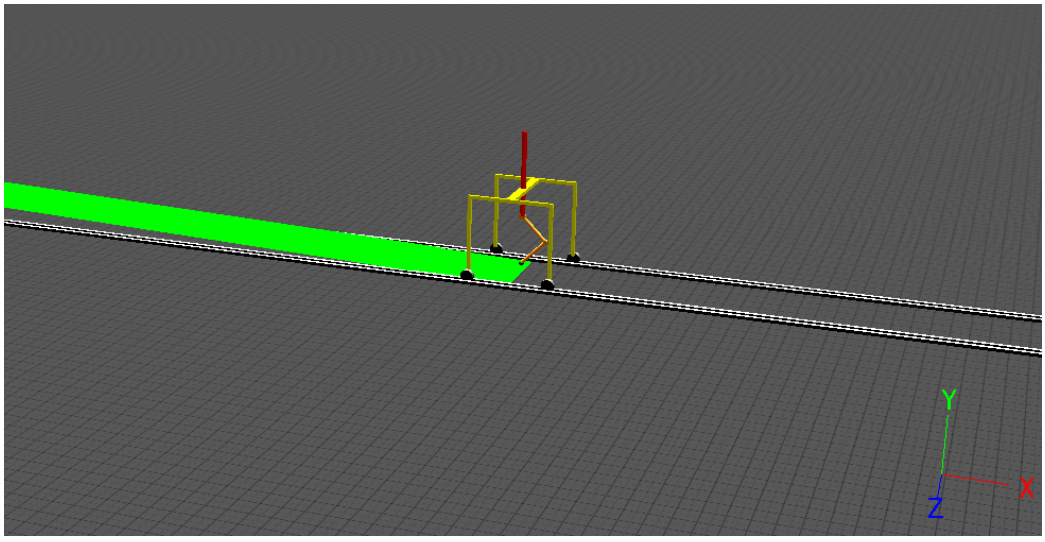


Figure 5.3: Simulated world on WEBOTS, for the seventh, eight and ninth experiments of the first stage.

The values of the mean velocity were calculated for each run. Afterwards, we set the value of T_{swing} at 0.9 seconds, which allowed us to plot the hip oscillator's amplitude against the mean velocity in a 2D plot.

5.2.1 Multiple linear regression approximation

In the next stage of this work, a forward model was used to control the parameters. A forward model uses states (s) and actions (u) to predict outcomes (z):

$$z = \hat{f}(s,u), \tag{5.1}$$

where $\hat{f}()$ is a approximation of $f()$, which is not known at the beginning of the task. The state vector will consist in a pair of variables identifying the current floor type the system is traversing and the system's lift mass. The system's output for this stage will be defined as the average velocity of a 25 seconds run on one of the scenarios, v_{avg}. For each scenario, the CPG network hip amplitude, μ_{hip} and swing phase period $T_{\text{swing,hip}}$, were evaluated for a predefined set of values.

In practice, both the state variables were assumed as known, and treated as discrete quantities, and $T_{\text{swing,hip}}$ was fixated before obtaining the approximated models, so we can obtain one model for each of the nine evaluated scenario's,

$$v_{avg} = \hat{f}(\mu_{\text{hip}}). \tag{5.2}$$

This way we can obtain a set of one dimensional models. Ideally, we would have a four

dimensional model, directly using the lift mass as a continuous parameter, keeping the floor type as a discrete parameter, and not fixating $T_{swing,hip}$. This multidimensional approach was used on the second stage of the work.

The approximation was made using multiple linear regression using least squares [7]. This type of regression makes use of p predictors and N observations, and returns a $p \times 1$ vector β of regression coefficients to be estimated in the linear model $Y = X\beta$. X is a $N \times p$ design matrix, Y is a $N \times 1$ vector of response observations, which consists on our previous experiments results. The predictor variables used for all experiments were 1, μ, μ^2, μ^3 and μ^4 . The plots of the experiments results, as well as the corresponding regression, are presented in figure 5.4.

5.3 Maintaining the mean velocity

After obtaining an approximation of the relation between mean velocity and the hip oscillator's amplitude on the leg, on the form

$$v_{avg} = b_1 + b_2 \times \mu + b_3 \times \mu^2 + b_4 \times \mu^3 + b_5 \times \mu^4 \tag{5.3}$$

where v_{avg} denotes de mean velocity, $b_1 \ldots b_5$ are the regression coefficients, and μ is the oscillator's amplitude, we isolate the variable μ in (5.3),

$$b_2 \times \mu + b_3 \times \mu^2 + b_4 \times \mu^3 + b_5 \times \mu^4 = v_{avg} - b_1 \tag{5.4}$$

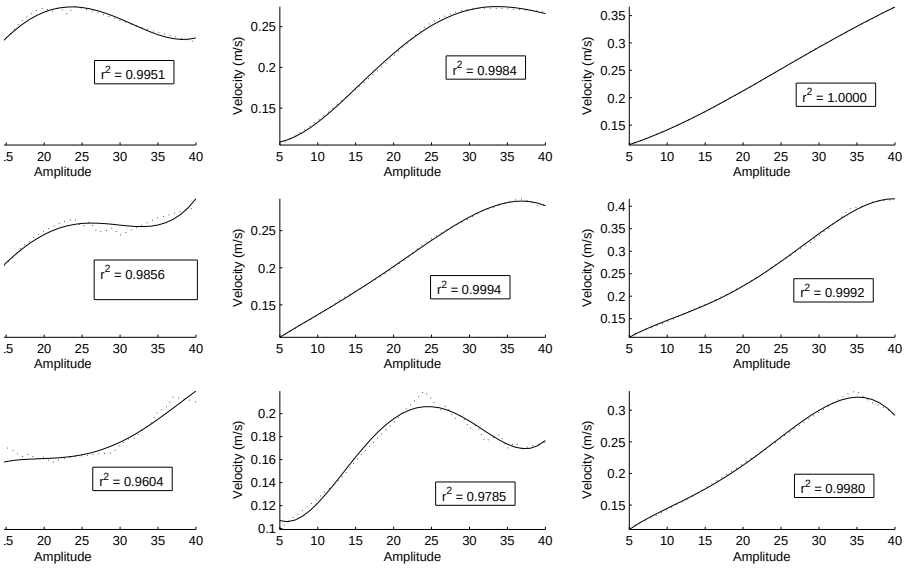


Figure 5.4: Speed characterization of the robot for different values of ω and $T_{swing} = 0.9$. The plots correspond to the experiments numbered from 1 to 9, in the order from left to right, and from top to bottom. The experiment observations are plotted in a dotted line, while the regression model is plotted in a solid line. The values of the r^2 statistic for each regression are also shown.

and then we can use Newton's method to find a solution for a specific mean velocity

$$b_2 \times \mu + b_3 \times \mu^2 + b_4 \times \mu^3 + b_5 \times \mu^4 - v_{avg,d} + b_1 = 0. \tag{5.5}$$

This is the equation we use to find the amplitude needed in order to maintain the desired velocity, $v_{avg,d}$.

The WEBOTS world used in this stage is shown in figure 5.5. The leg was set in motion for 120 seconds, and went through the three different floor types, with the weight of the lift changing in set points in time. The weight started at 1Kg and was changed to 10Kg at the 30s mark, 5Kg at 40s, 1Kg at 70s, 10Kg at 90s, and 5Kg at 110s. The oscillator's amplitude was changed every time the lift mass or the floor type changed, using (5.5), with the regression coefficients corresponding to the experiment which used the current lift mass/floor type combination, and a desired velocity of 0.15 m/s.

5.4 Results

The mean velocity achieved on this stage was of 0.1507 m/s, against the desired 0.15 m/s. Figure 5.6 shows the evolution of the mean velocity of the leg through the whole run. We can see that the mean velocity shows a more or less steady increase until it reaches a value of approximately 0.15 m/s (the desired value) at around 12 seconds of the experiment,and remains almost constant afterwards. The hip oscillator's amplitude changes with each lift mass/floor type change. The floor type changes twice in a short period of time at around 35s and 70s due to the nature of the leg's walking motion swing phase, which as a small

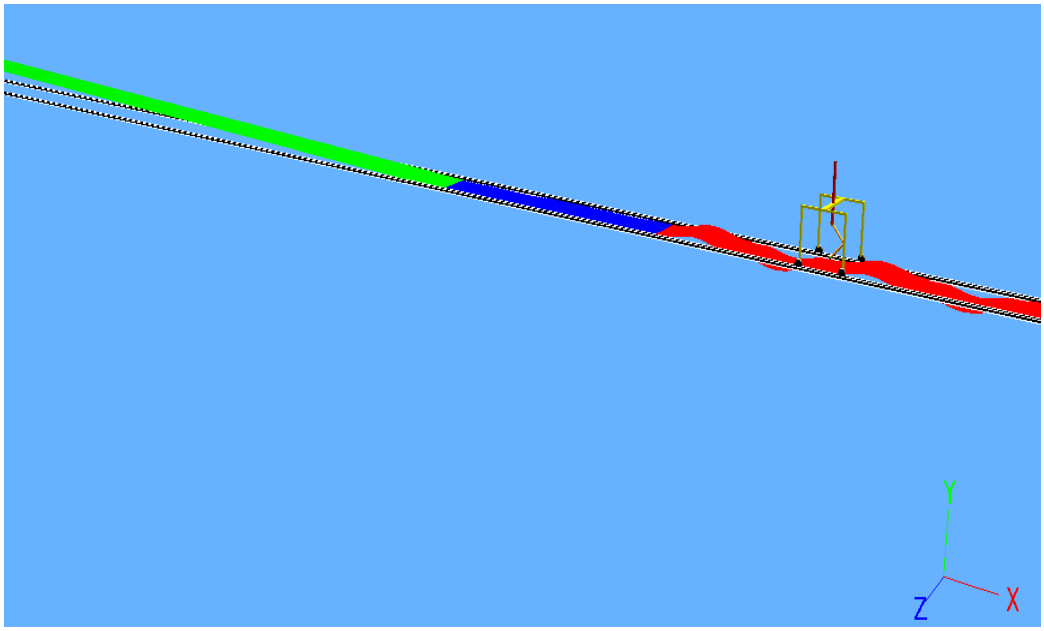


Figure 5.5: Simulated world on WEBOTS, for the second stage.

period where the leg first retracts, so it can propel forward afterwards.

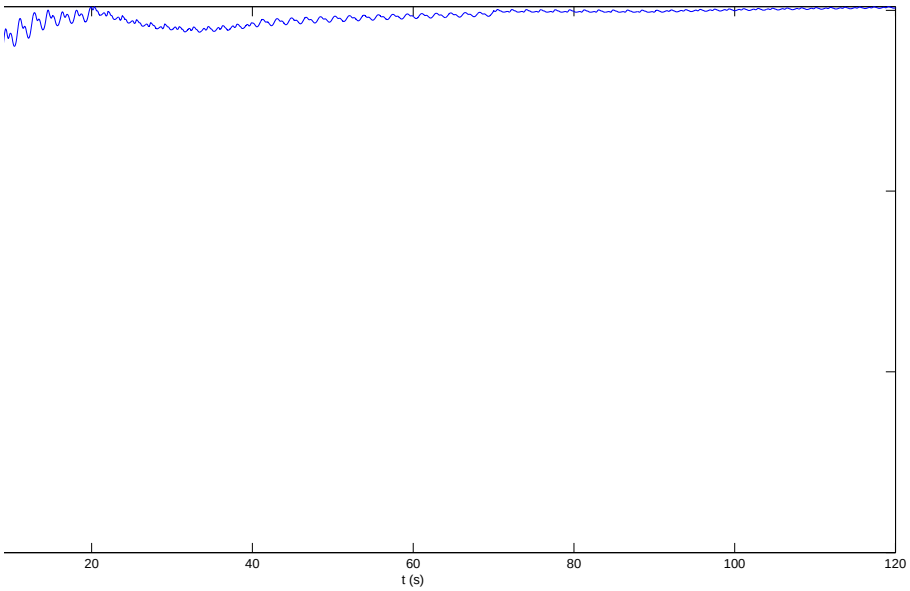


Figure 5.6: Mean velocity against time, for the second stage.

Chapter 6

Velocity control using an inverse model

The inverse model approach, introduced in chapter 4, section 4.3, was used in the second stage of this work.

As stated before, and inverse model uses states (s) and outcomes (z) to predict the necessary action (u) [5]

$$u = \hat{f}^{-1}(s, z), \tag{6.1}$$

In this approach, the output is the average velocity of each stride of the system's locomotion. The state vector will contain the total displacement of the system's center of mass in the

last stride, Δ_x, and the mean ground reaction force of the system's lower end along the simulated world x axis direction of the last stride, GRF_x. The action chosen on this stage is the CPG network hip's swing phase duration, $T_{swing,hip}$. The reason to evaluate the system at each stride as to pertain with the periodic nature of the control system. The position of the joints is controlled throughout each stride by the control network output, and while we don't control the position at each simulation time step, we can control the parameters that mold the sinusoidal that will control the system for each stride. As such, we focus on learning the correct parameters for the next stride, to reach the desired speed. This approach is possible, as opposed to the evaluation on a longer time scale that was done in the first stage, because the learning algorithm we used for the inverse model uses incremental and online learning.

The learning algorithm used in this stage was Locally Weighted Projection Regression [41]. This algorithm achieves nonlinear function approximation through the use of various local linear models, that have associated weighting kernels which are used to give each model the correct contribution to the output of a query point. Each model, called receptive field, projects the input vector on the most relevant dimensions to estimate the output vector. This small number of univariate regressions in selected directions is achieved using a modified version of Partial Least Squares, while the receptive fields are learned through local regression. Both this learning steps are done incrementally, which allows the size and shape of the receptive fields to change both during training and during testing, this feature being the main reason this algorithm was chosen for the task.

The inverse model needed to control the system's traveling speed can be described as

$$T_{\text{swing,hip}} = \hat{f}^{-1}(v_{avg}, \Delta_x, GRF_x), \tag{6.2}$$

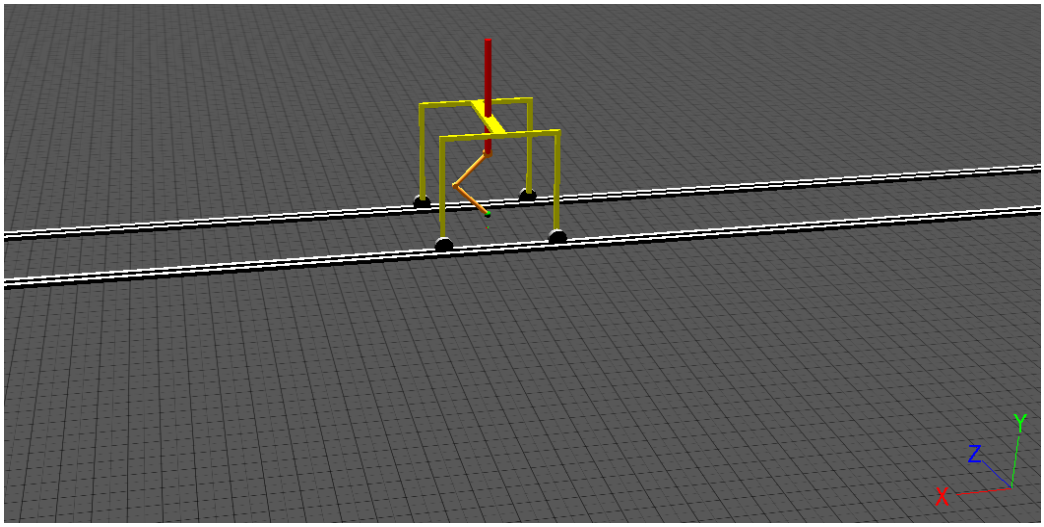


Figure 6.1: Simulated environment used in the first training and first and second sets of tests for this stage of the work.

where v_{avg} is redefined from the previous chapter. This model was obtained through the use of LWPR by running the system in a single run, on the simulated environment of Figures 6.1 and 6.2 changing the $T_{\text{swing,hip}}$ parameter at an interval of two strides. The obtained model was then tested on simulated environments with different floor conditions and different lift masses, compared to the training phase. This consists in a training phase with much less data used, compared to the first stage, was well as no training in different floor types and with different lift masses. This was done with the intention of showing the online adaptive capabilities of LWPR.

The simulated environments were the velocity control of the system was tested are showed on Figures 6.1 and 6.3.

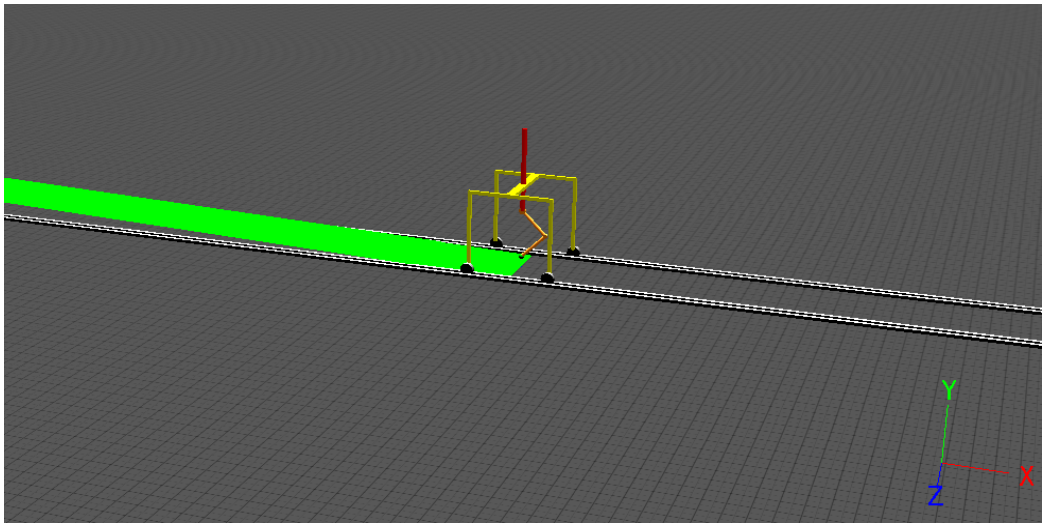


Figure 6.2: Simulated environment used in the second training for this stage of the work.

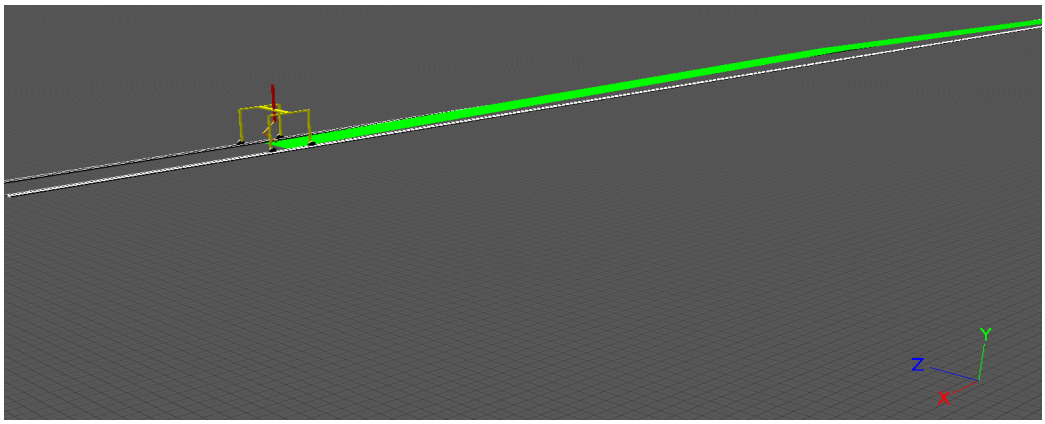


Figure 6.3: Simulated environment used in the third set of tests for this stage of the work.

6.1 Training in the simulated environment

The system was trained in two separate training phases. It was ran in the simulated environments previously mentioned, controlled by a network of two CPGs, implemented through Hopf oscillators. The sinusoidal waves of the output of the x variable of the oscillators are translated into position commands for the actuated joints, resulting in a smooth locomotion.

The oscillator's parameters, described in chapter 4, sections 4.2, used for both the training phases were as follows: $\alpha_{hip} = \frac{1}{45}$ and $\alpha_{ankle} = \frac{1}{45}$, variables that control the speed of convergence for the limit cycle; $\mu_{hip} = 15 \times 15$ and $\mu_{ankle} = 15 \times 15$, where $\sqrt{\mu}$ represents the amplitude of the oscillations; the offsets where all set to 0; $\lambda = 0.5$, the duty factor, meaning the swing and the stance phases have the same duration; $b = 500$ controls the alternation speed of the frequency modulation.

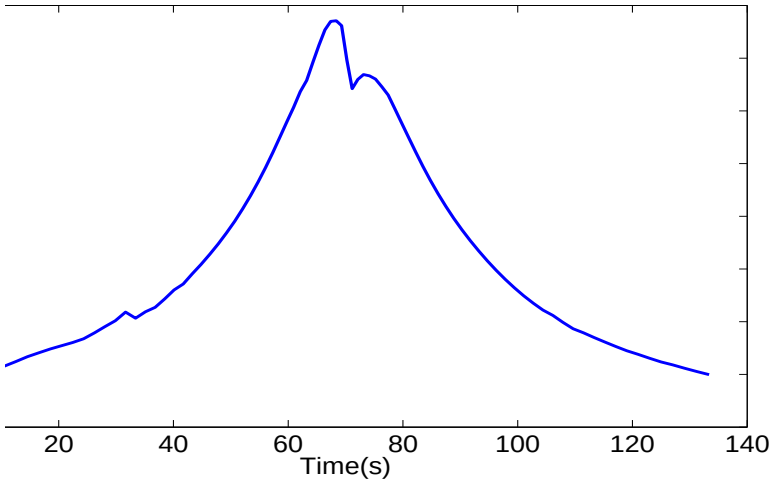
The T_{swing} parameter for the hip's oscillator was the one chosen as the control variable of the learning control approach. The system's velocity is closely related to the period of both phases of the oscillator cycle, as stated on chapter 4. The T_{swing} of the ankle's oscillator was set as twice the value of that of the hip's oscillator. To evaluate the system's response to various values of T_{swing}, the system was ran during 135 seconds of experiment, with the first 5 not accounted for, to give time for the oscillator to reach a limit cycle and the system's general locomotion to stabilize. During those $135s$, the T_{swing} was started at $1.085s$, reduced to $0.455s$ at midpoint of the $130s$, and again raised to $1.085s$ by the end of the simulation. The parameter was changed with $0.015s$ increments, with each change occurring at an interval of two strides of the locomotion.

The first training phase was performed on the environment presented in Figure 6.1. The

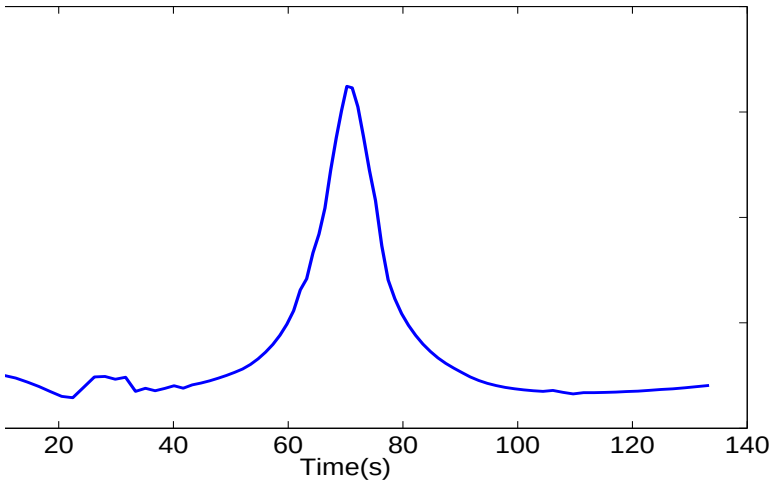
average velocity and the GRF in the x axis observed through the training can be seen in Figure 6.4. The COM displacement Δ_x, is not presented because of how closely related it is to the average velocity. As expected, the average velocity steadily until the midpoint of the simulation, coincidently with the descent of the the swing phase period, and conversely drops after that. The GRF follows a similar pattern, but has a more accentuated change in the midpoint of the training, where the swing phase duration is shorter. This is likely due to a less stable locomotion caused by having very short strides, which is backed up by the fact that the average velocity values show an irregularity in its descent after the midpoint. Lower values of the swing phase period were not used precisely because they could cause an extremely unstable locomotion, at times resulting in the system falling over.

6.2 Obtaining the LWPR model

The training data was then used to obtain a model of equation 6.2, by using LWPR. A model was obtained for each trail of a velocity control test, with three trials per velocity value, and with three values of velocities tested. Table 6.1 shows some statistics pertaining to the obtained models. The majority of the obtained approximations used 3 receptive fields to describe the behavior observer, related to the variables used from equation 6.2. This approximations all showed training normalized mean squared errors between 0.003 and 0.006, and between 0.007 and 0.011 during testing. These lead to believe the approximations were a good representation of the intended model. Figure 6.5 show the the predictions of the models and the actual output of the data, for the LWPR test points of one of the approximations. One can see that the predictions come very close to the collected data points, which are, to some extent, encapsulated by the confidence bounds. Similar

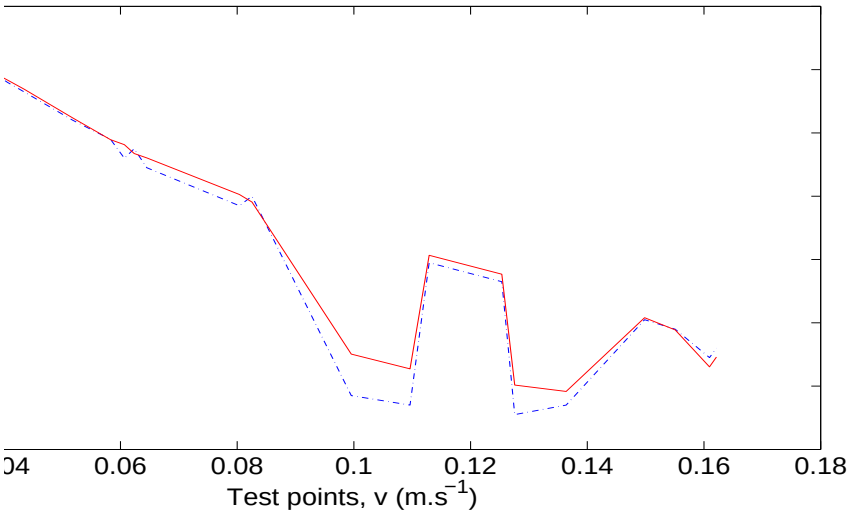


(a) Average velocity.

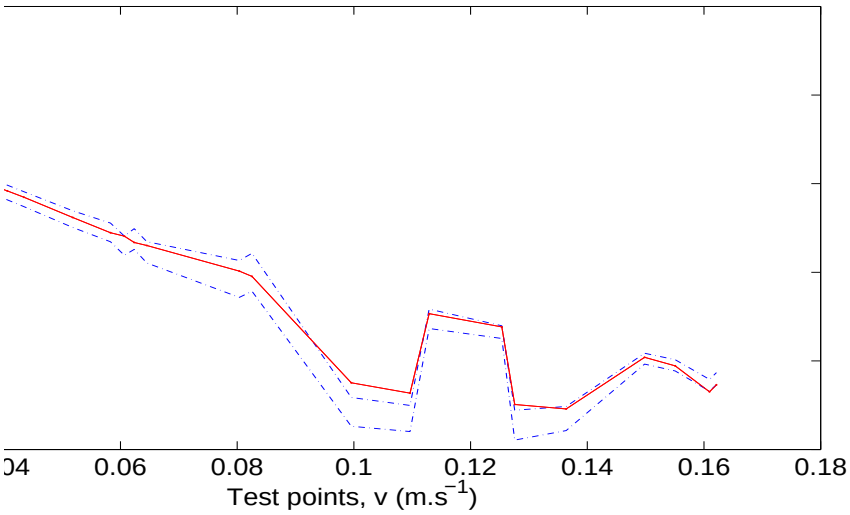


(b) Ground reaction force, x axis.

Figure 6.4: Data from the first training phase used in the regression.



(a) Test points and the regression predictions.



(b) Test points and the regression confidence bounds.

Figure 6.5: Example regression test set results for the first training phase. The test data points are the desired output (v_{avg} in $m.s^{-1}$), with the corresponding control point ($T, inseconds$). The solid red line shows the data collected during training, and the blue dashed line shows a) the predictions and the b) confidence bounds.

results were observed by the other approximations.

The obtained approximations were used to control the hopper in manner similar to the training simulation. The parameters used were the same, except for $T_{swing,hip}$. This parameter was adjusted between every two strides, by calculating the necessary value to achieve a specific average velocity with LWPR model. The model was updated with each point of the test run, which provides the capability of online learning to the control, and the potential to adapt to changes to the environment, the locomotion system, and the interactions between each other, during run time. The hopper was controlled for velocities of $0.06m.s^{-1}$, $0.09m.s^{-1}$ and $0.12m.s^{-1}$, with three trials for each velocity. The first batch of training data was used to control the hopper in the environment showed in Figure 6.1. This environment was used for two sets of tests, with each set consisting of an evaluation of the three velocities. The first set used the same exact conditions as the first training set. The second used different values of mass for the lift (the uppermost segment) of the system throughout each run. The mass of the hopper's lift started at 5 Kg, the value used during training, and was changed to 8 Kg, 11 Kg, 3 Kg and then back to 5 Kg, with the changes occurring every 40 seconds. The first set of test runs were simulations with the duration of 105 seconds, and the second of 200 seconds.

6.3 Results for the control with the first model

The goal was to maintain the system's average velocity of each stride as close as possible to the desired values, with the same environment conditions between the training and testing phases, and different lift mass values throughout one set of test runs, as opposed to a

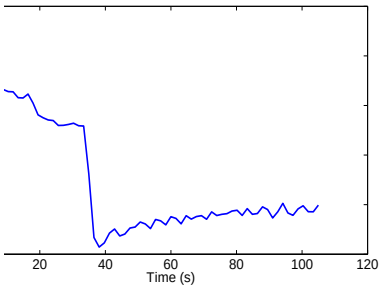
Table 6.1: Results for the first set of test runs.

desired v_{avg}		LWPR model				v_{avg} obtained ($m.s^{-1}$)	
($m.s^{-1}$)		#Samples	#RFs	nMSE	nMSE (testing)	mean	$\pm$ $s.d.$
0.06	trial 1	1770	3	0.004	0.007	0.0620	0.0054
	trial 2	1770	3	0.003	0.007	0.0610	0.0047
	trial 3	1770	3	0.003	0.009	0.0629	0.0095
0.09	trial 1	1770	2	0.006	0.011	0.0919	0.0038
	trial 2	1770	3	0.004	0.010	0.0906	0.0073
	trial 3	1770	3	0.004	0.009	0.0910	0.0050
0.12	trial 1	1770	3	0.003	0.009	0.1208	0.0028
	trial 2	1770	3	0.004	0.010	0.1203	0.0019
	trial 3	1770	3	0.004	0.009	0.1206	0.0025

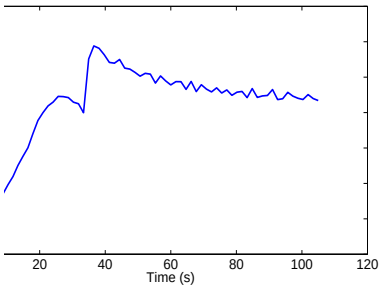
value of 5 Kg during the whole training run.

The results obtained for the first test run can be seen in Figures 6.6, 6.7 and 6.8. The mean and standard deviations of the average velocity for each run are presented in Table 6.1. As expected by running the tests in the same conditions used in the training, and with good approximations from the LWPR models, the results show a mean close to the target velocities with relatively small variation. The velocity was achieved and stabilized, in most cases, around $40s$ of the simulation, with the second trial for $0.12m.s^{-1}$ being a notable exception, with the velocity still oscillating at the end of the simulation, as we can see in Figure 6.8 c). Comparing the results of the average velocity with those of the swing phase period we can verify that they show variations at the same points in time, which is expected because no perturbations where introduced relatively to the training phase, and the only parameter being changed is the swing period. The second set of testing of the velocity control introduced the changes in mass to test for the ability to adapt to such perturbations.

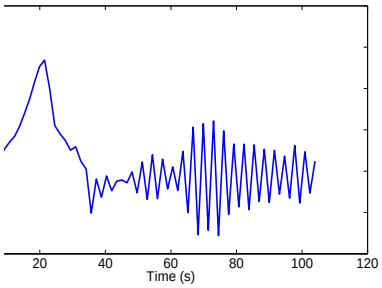
Figures 6.9, 6.10 and 6.11 show the results for the second phase of testing, along with Table 6.2. The means and standard deviations of this set of tests shows a greater variation from the desired results, when compared to those of the first set. This variation is largely due to accentuated changes at the points were the mass is changed, particularly with the steep change from 11 Kg to 3 Kg (the third vertical dashed red line in the average velocity plot marks this transition). In most cases the controller manages to adapt itself to the correct velocity after at most 20 seconds, adapting a lot faster than that in some situations. There are some exceptions where the hopper's velocity oscillates during the entire period of time designated for a specific velocity, as happens with the 3 Kg phase in Figure 6.11 e).



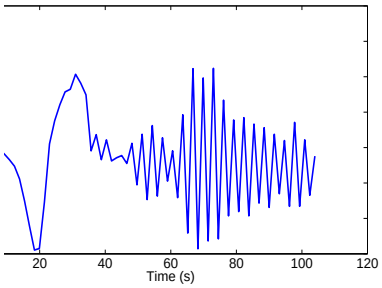
(a) Average velocity, first trial.



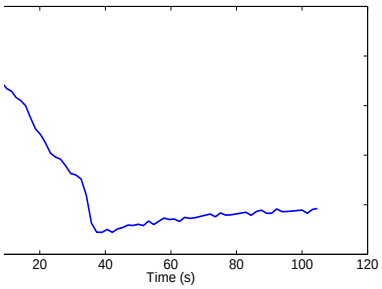
(b) Hip's CPG swing period, first trial.



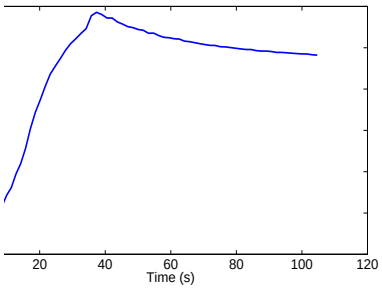
(c) Average velocity, second trial.



(d) Hip's CPG swing period, second trial.

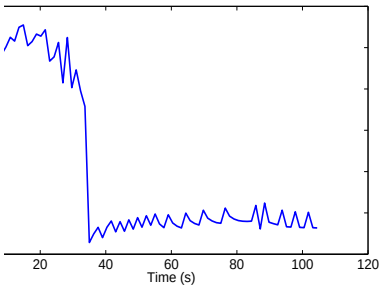


(e) Average velocity, third trial.

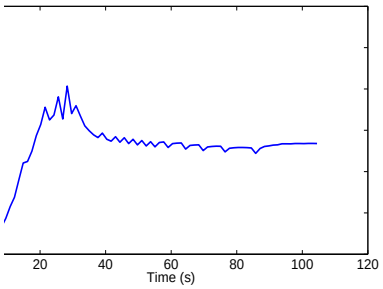


(f) Hip's CPG swing period, third trial.

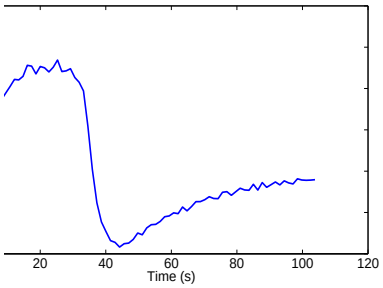
Figure 6.6: Output (v_{avg}) and control (T_{swing}) variables against time, for the first control test, desired v_{avg} of $0.06\ m.s^{-1}$, all three trials.



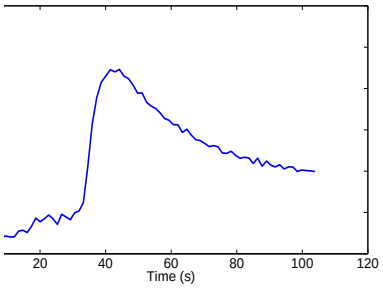
(a) Average velocity, first trial.



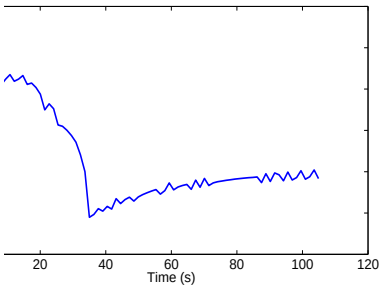
(b) Hip's CPG swing period, first trial.



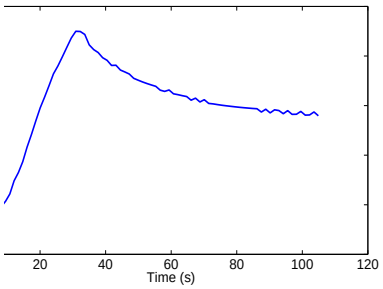
(c) Average velocity, second trial.



(d) Hip's CPG swing period, second trial.

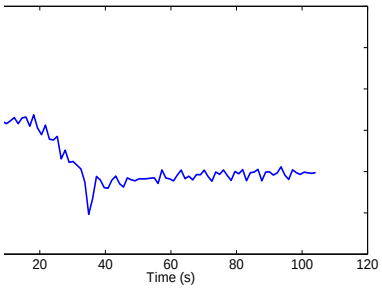


(e) Average velocity, third trial.

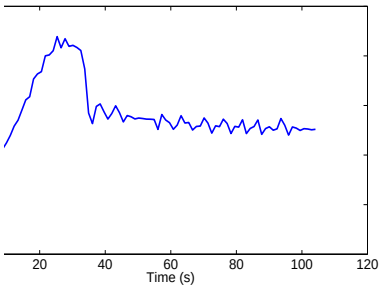


(f) Hip's CPG swing period, third trial.

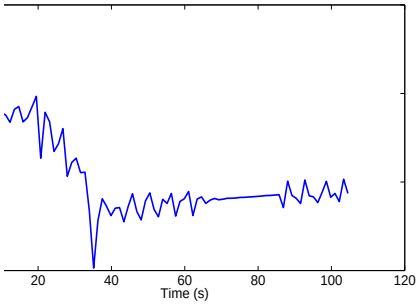
Figure 6.7: Output (v_{avg}) and control (T_{swing}) variables against time, for the first control test, desired v_{avg} of $0.09\ m.s^{-1}$, all three trials.



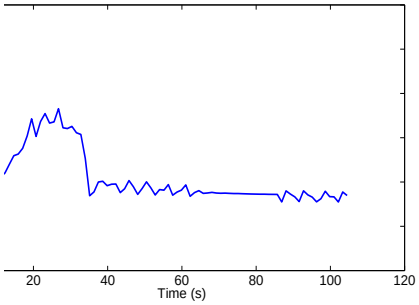
(a) Average velocity, first trial.



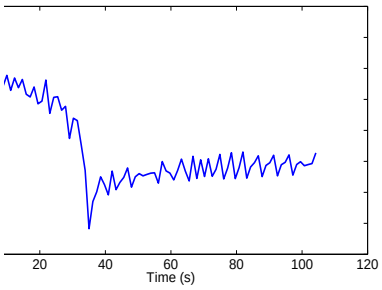
(b) Hip's CPG swing period, first trial.



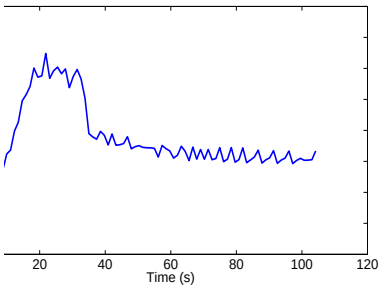
(c) Average velocity, second trial.



(d) Hip's CPG swing period, second trial.



(e) Average velocity, third trial.

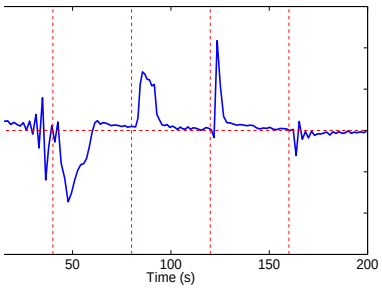


(f) Hip's CPG swing period, third trial.

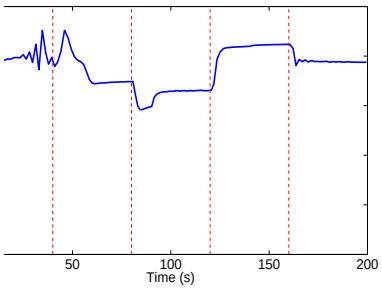
Figure 6.8: Output (v_{avg}) and control (T_{swing}) variables against time, for the first control test, desired v_{avg} of $0.12\ m.s^{-1}$, all three trials.

Table 6.2: Results for the second set of test runs.

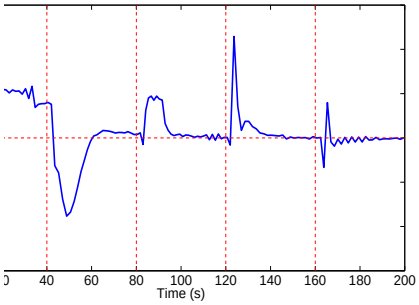
desired v_{avg}		LWPR model				v_{avg} obtained ($m.s^{-1}$)	
($m.s^{-1}$)		#Samples	#RFs	nMSE	nMSE (testing)	mean	$\pm$ $s.d.$
0.06	trial 1	1770	3	0.004	0.009	0.0612	0.0100
	trial 2	1770	2	0.006	0.014	0.0641	0.0118
	trial 3	1770	3	0.005	0.011	0.0649	0.0129
0.09	trial 1	1770	2	0.005	0.010	0.0928	0.0095
	trial 2	1770	3	0.006	0.016	0.0915	0.0143
	trial 3	1770	3	0.007	0.014	0.0949	0.0132
0.12	trial 1	1770	3	0.004	0.011	0.1176	0.0159
	trial 2	1770	3	0.004	0.011	0.1173	0.0139
	trial 3	1770	3	0.003	0.007	0.1133	0.0506



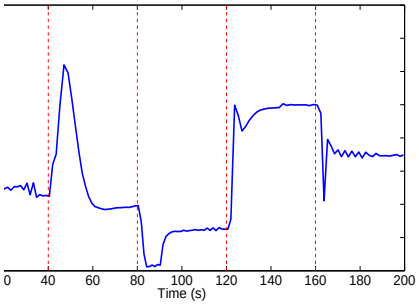
(a) Average velocity, first trial.



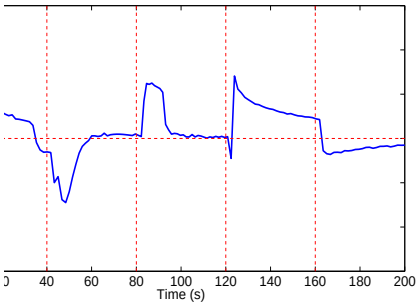
(b) Hip's CPG swing period, first trial.



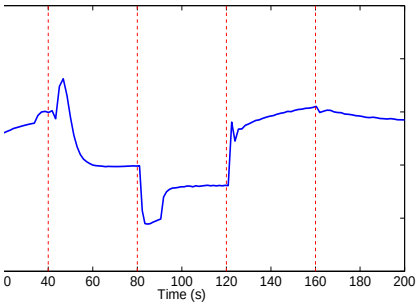
(c) Average velocity, second trial.



(d) Hip's CPG swing period, second trial.

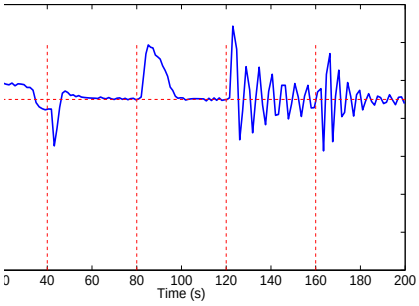


(e) Average velocity, third trial.

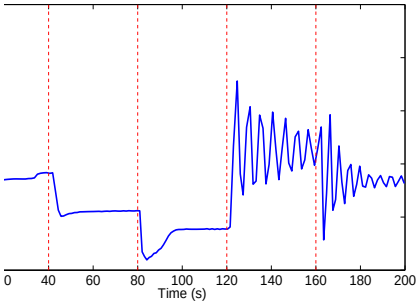


(f) Hip's CPG swing period, third trial.

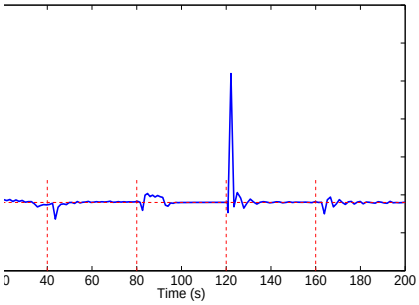
Figure 6.9: Output (v_{avg}) and control (T_{swing}) variables against time, for the second control test, desired v_{avg} of $0.06\ m.s^{-1}$, all three trials. The solid blue lines show the values for v_{avg} and T_{swing}. The vertical dashed red lines show the points in time where a change of mass occurred, while the horizontal show the desired velocity for the test.



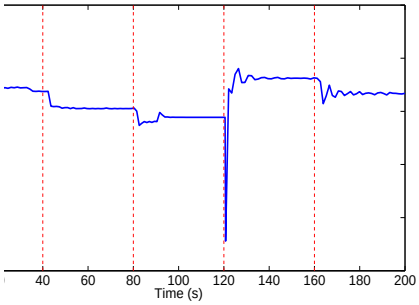
(a) Average velocity, first trial.



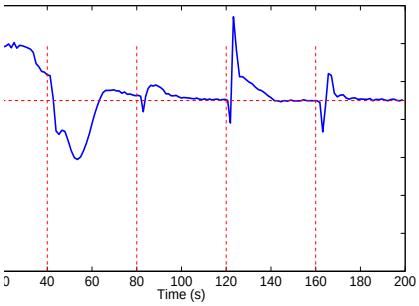
(b) Hip's CPG swing period, first trial.



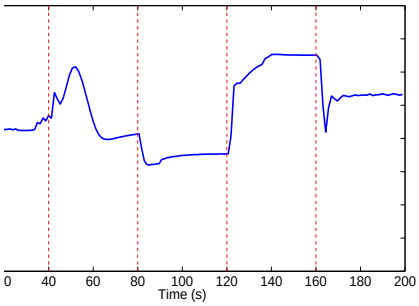
(c) Average velocity, second trial.



(d) Hip's CPG swing period, second trial.

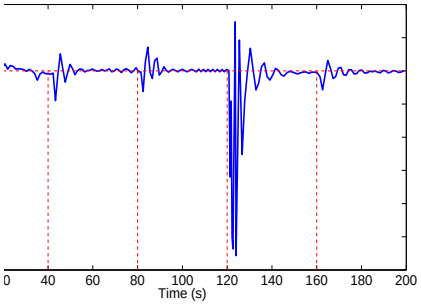


(e) Average velocity, third trial.

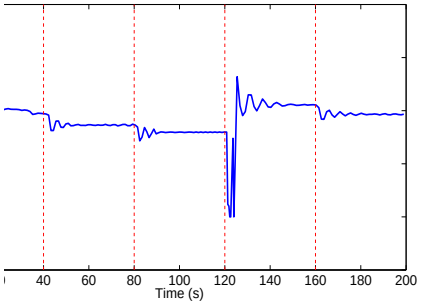


(f) Hip's CPG swing period, third trial.

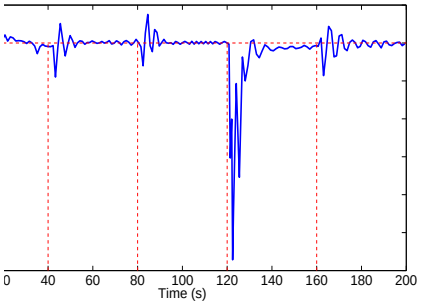
Figure 6.10: Output (v_{avg}) and control (T_{swing}) variables against time, for the second control test, desired v_{avg} of $0.09\ m.s^{-1}$, all three trials. The solid blue lines show the values for v_{avg} and T_{swing}. The vertical dashed red lines show the points in time where a change of mass occurred, while the horizontal show the desired velocity for the test.



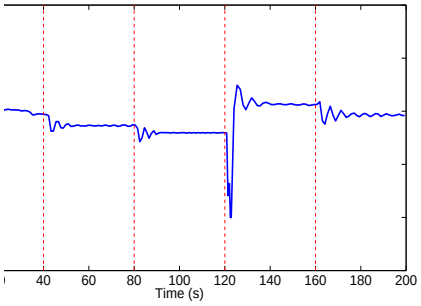
(a) Average velocity, first trial.



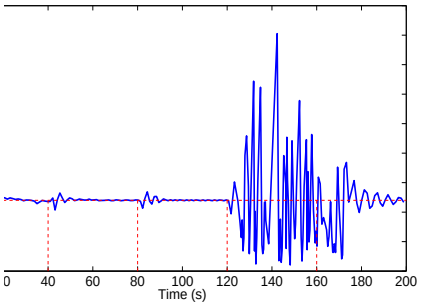
(b) Hip's CPG swing period, first trial.



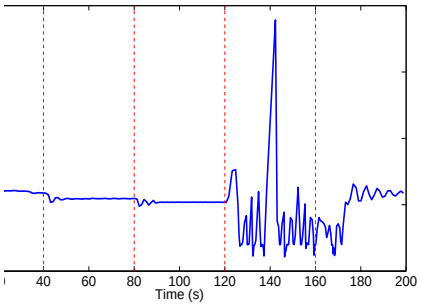
(c) Average velocity, second trial.



(d) Hip's CPG swing period, second trial.



(e) Average velocity, third trial.



(f) Hip's CPG swing period, third trial.

Figure 6.11: Output (v_{avg}) and control (T_{swing}) variables against time, for the second control test, desired v_{avg} of $0.12\ m.s^{-1}$, all three trials. The solid blue lines show the values for v_{avg} and T_{swing}. The vertical dashed red lines show the points in time where a change of mass occurred, while the horizontal show the desired velocity for the test.

6.4 Velocity control while transversing a ramp

The second training run to collect data for the LWPR models was done with the addition of a ramp to the first world, as can be seen in Figure 6.2. With the addition of terrain height changes, two more control variables were introduced to the model in equation 6.2,

$$T_{\text{swing,hip}} = \hat{f}^{-1}(v_{avg}, \Delta_x, GRF_x, GRF_y, lower_y). \tag{6.3}$$

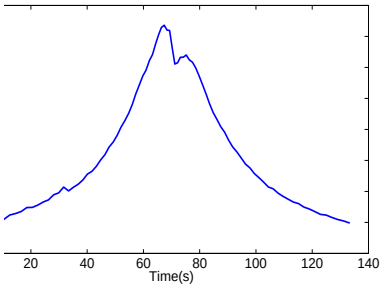
Here, GRF_y is the mean ground reaction force caused by the contact of the hopper with the ground, in the y axis, and $lower_y$ is the lowest height point attained by the hip joint during the last stride of the locomotion. This changes have the purpose of accounting for the changes in height the systems goes through, and giving it the ability to adapt correctly to these changes. The parameters used during the training were the exact same ones used in Section 6.2. The average velocity, GRF in the x and y axis, and the lowest position achieved on the y axis by the hopper during this training run are presented in Figure 6.12. The average velocity and the GRF in the x axis have a similar behavior to those observed in the data from the first training run, with lower values of velocity achieved, overall. The lowest height of the hip joint increases throughout the run, due to the fact that the ramp introduced in the environment as a constant upward slope.

This training data was used to train LWPR models to be used in a third set of tests of velocity control. An example of the test data for the models, the model's predictions, and its confidence bounds plots are presented in Figure 6.13. Similarly to the models obtained in the first round of LWPR models, the outputs match the predictions closely, and are mostly

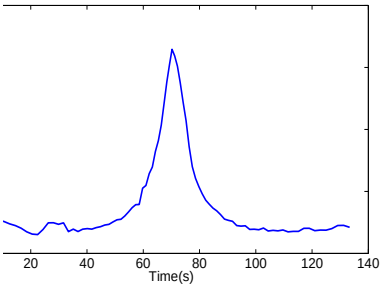
contained in the confidence bounds. Table 6.3 shows the number of training data evaluated by the models, the number of receptive fields of each model, and the normalized mean square error during training and testing of the models. The errors observed were again very low, but the number of receptive fields needed to correctly approximate the model raised from 2 or 3 models to 5 or 6 models.

The LWPR models obtained were used to control the hopper's velocity in the same manner as with the previous test sets. Again the velocity was controlled for $0.06\,m.s^{-1}$, $0.09\,m.s^{-1}$ and $0.12 m.s^{-1}$, with three trials for each velocity. The environment used contained a ramp which begins with an upward slope, and then, afters its midpoint, proceeds with a downward slope with the same inclination, as observed in Figure 6.3. Each simulation was ran for 200 seconds.

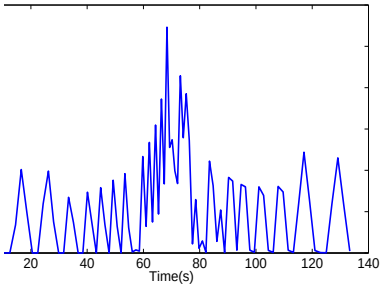
The results for each test run are presented in Table 6.3 (mean average velocity and its standard deviation) and in Figures 6.14, 6.15, and 6.16. The mean and standard deviation values showed less variation than the second test set, with values almost as good as the first set. To evaluate the plots, we first must consider how far on the ramp the hopper managed to get in each run. As the mean values closely follow the desired ones, we can assume the same reaching for each set of trials for on determined velocity. With $0.06\ m.s^{-1}$ the hopper got to about two thirds of the ramp, with $0.09\ m.s^{-1}$ the hopper ended at the end of the ramp, and with a velocity of $0.12\ m.s^{-1}$ the hopper went on further than the ramp, along a leveled floor. For the runs controlled to $0.06\ m.s^{-1}$ the velocity begins at a value to high, overshoots under the desired value at around 40 seconds, and after a period of stabilization overshoots the value again, with two of the trials showing sign of stabilization thereafter. The second overshoot is most likely due to the hopper entering the downward slope of the ramp, raising its velocity due to the effect of gravity. The tests for $0.09\ m.s^{-1}$



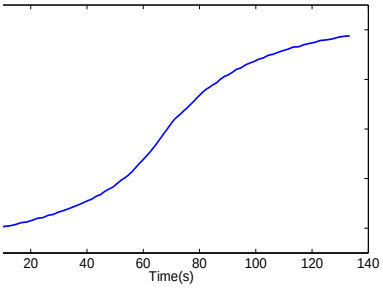
(a) Average velocity.



(b) Ground reaction force, x axis.



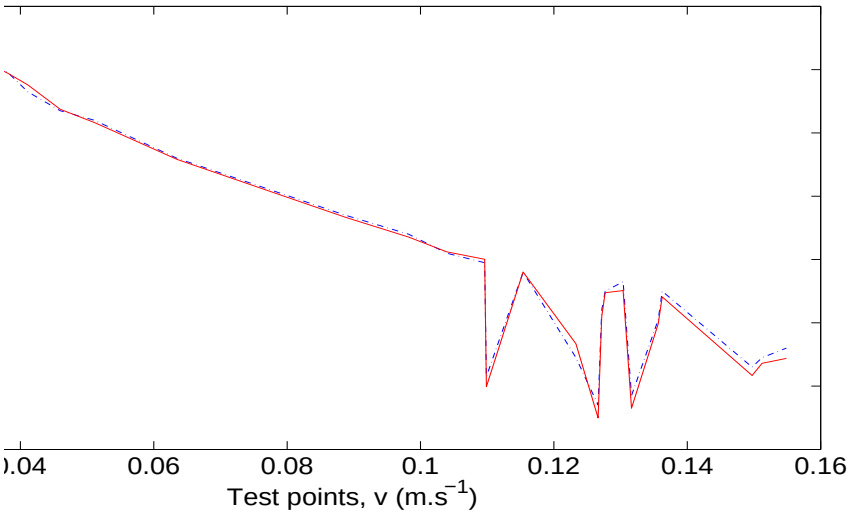
(c) Ground reaction force, y axis.



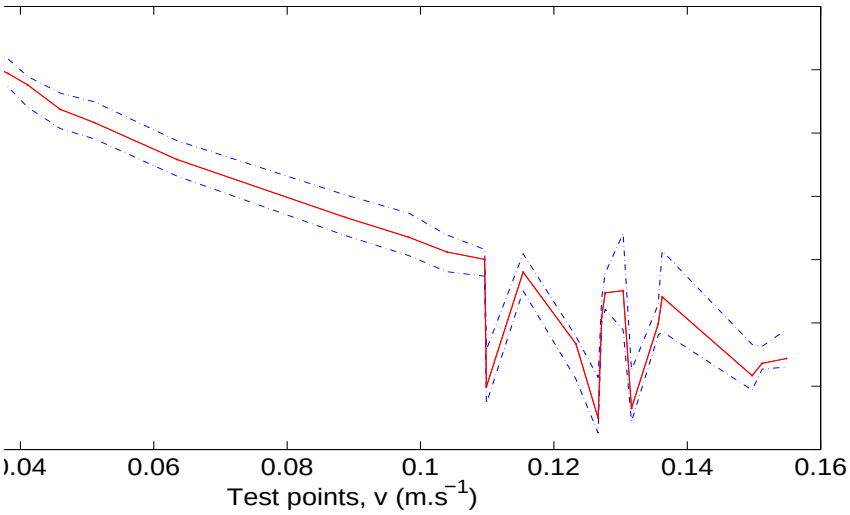
(d) Lowest position achieved by the hip joint, y axis.

Figure 6.12: Data from the second training phase used in the regression.

show a similar behavior, with the second overshoot occurring earlier, as the hopper reaches the midpoint of the ramp sooner because it moves faster. The fact that it gets to the end of the ramp allows it to a better stabilization than in the previous cases. The third and final set of trials, for $0.12\ m.s^{-1}$ show an even earlier second overshoot, and comprehensively an earlier stabilization than before. It is also important to mention that in this case the variations of the velocity do not always match with the variations of the swing phase period, as the perturbation caused by the inclination of the ramp is an external cause that modifies the velocity, and its something that is not directly incorporated in the model, but something it has to adapt to.



(a) Test points and the regression predictions.

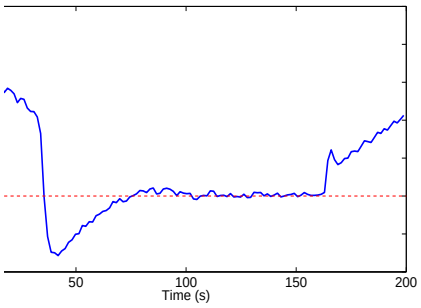


(b) Test points and the regression confidence bounds.

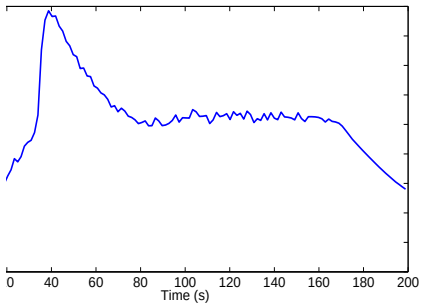
Figure 6.13: Example regression test set results for the second training phase. The test data points are the desired output (v_{avg} in $m.s^{-1}$), with the corresponding control point ($T, inseconds$). The solid red line shows the data collected during training, and the blue dashed line shows a) the predictions and the b) confidence bounds.

Table 6.3: Results for the third set of test runs.

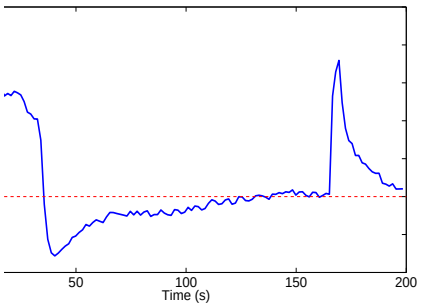
desired v_{avg}		LWPR model				v_{avg} obtained ($m.s^{-1}$)	
($m.s^{-1}$)		#Samples	#RFs	nMSE	nMSE (testing)	mean	$\pm$ $s.d.$
0.06	trial 1	1770	6	0.003	0.009	0.0629	0.0061
	trial 2	1770	6	0.004	0.007	0.0619	0.0064
	trial 3	1770	6	0.004	0.006	0.0620	0.0049
0.09	trial 1	1770	6	0.003	0.007	0.0925	0.0062
	trial 2	1770	5	0.002	0.006	0.0930	0.0031
	trial 3	1770	5	0.004	0.006	0.0927	0.0063
0.12	trial 1	1770	5	0.003	0.011	0.1234	0.0063
	trial 2	1770	5	0.003	0.007	0.1217	0.0039
	trial 3	1770	5	0.005	0.009	0.1224	0.0061



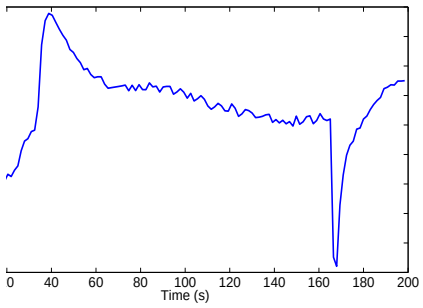
(a) Average velocity, first trial.



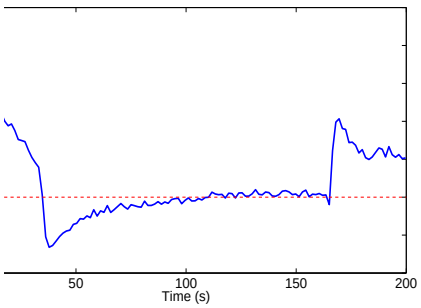
(b) Hip's CPG swing period, first trial.



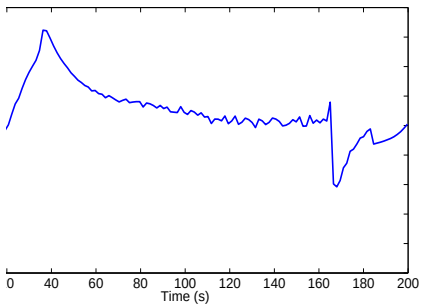
(c) Average velocity, second trial.



(d) Hip's CPG swing period, second trial.

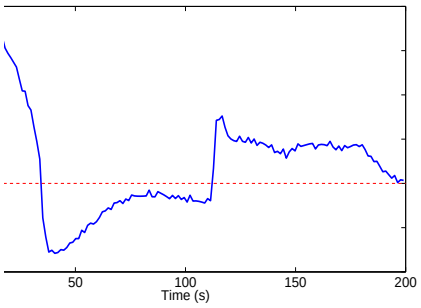


(e) Average velocity, third trial.

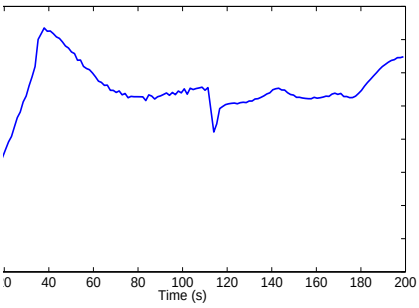


(f) Hip's CPG swing period, third trial.

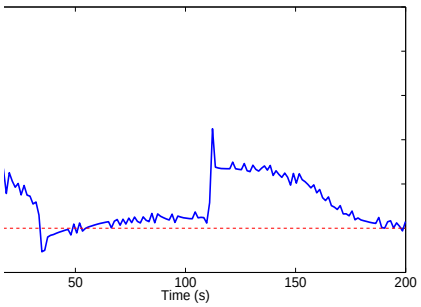
Figure 6.14: Output (v_{avg}) and control (T_{swing}) variables against time, for the third control test, desired v_{avg} of $0.06\ m.s^{-1}$, all three trials. The solid blue lines show the values for v_{avg} and T_{swing}. The horizontal dashed red line show the desired velocity for the test.



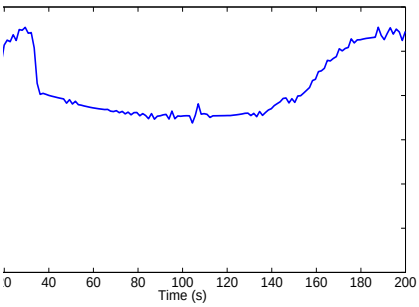
(a) Average velocity, first trial.



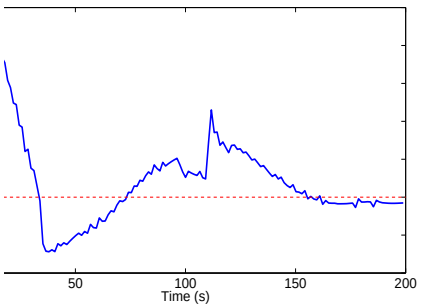
(b) Hip's CPG swing period, first trial.



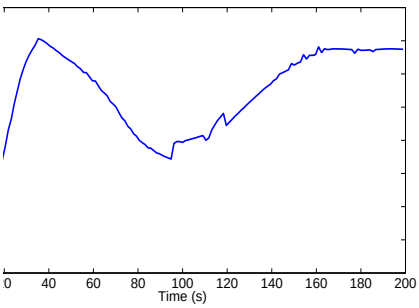
(c) Average velocity, second trial.



(d) Hip's CPG swing period, second trial.

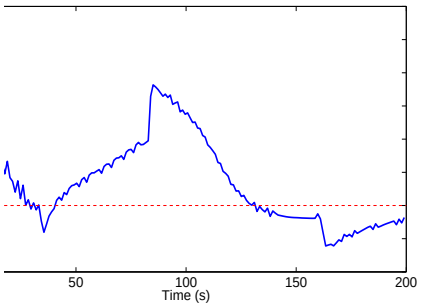


(e) Average velocity, third trial.

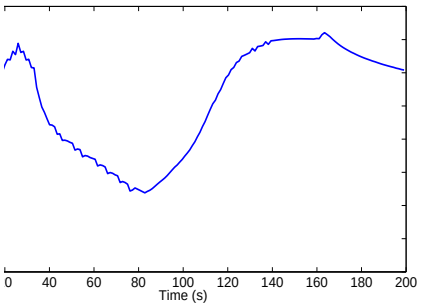


(f) Hip's CPG swing period, third trial.

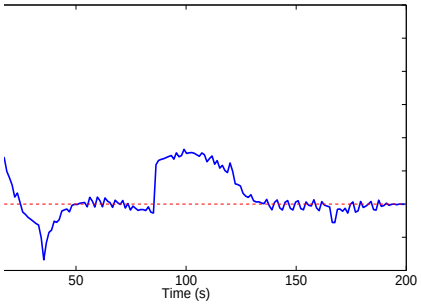
Figure 6.15: Output (v_{avg}) and control (T_{swing}) variables against time, for the third control test, desired v_{avg} of $0.09\ m.s^{-1}$, all three trials. The solid blue lines show the values for v_{avg} and T_{swing}. The horizontal dashed red line show the desired velocity for the test.



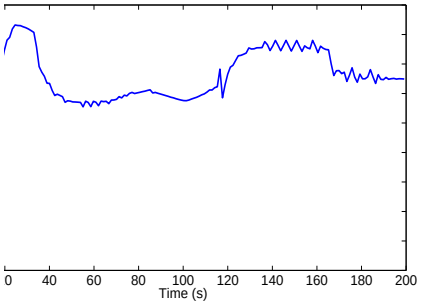
(a) Average velocity, first trial.



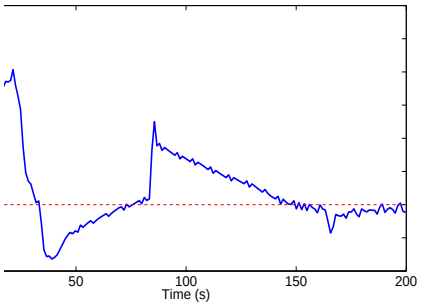
(b) Hip's CPG swing period, first trial.



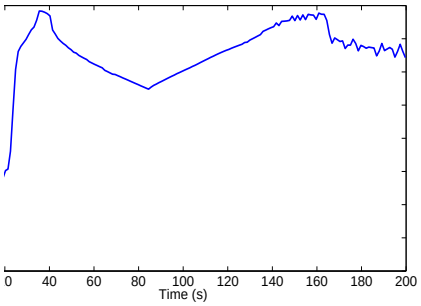
(c) Average velocity, second trial.



(d) Hip's CPG swing period, second trial.



(e) Average velocity, third trial.



(f) Hip's CPG swing period, third trial.

Figure 6.16: Output (v_{avg}) and control (T_{swing}) variables against time, for the third control test, desired v_{avg} of $0.12\ m.s^{-1}$, all three trials. The solid blue lines show the values for v_{avg} and T_{swing}. The horizontal dashed red line show the desired velocity for the test.

Chapter 7

Conclusions

The Multiple Linear Regression stage experiments, essential for the second stage, where also very lengthly and of a great number. More importantly, they are not generalized. A different lift mass can't be evaluated from the obtained equations as they are, without first running the experiment for the first stage for that specific mass, and the same can be said for a ramp with a different inclination, for example. A larger training set in the Locally Weighted Projection Regression would certainly improve the control outcome, but it was purposely left as small as possible in order to ensure that the approach could be used with minimal computation time and effort from the user.

This experiments showed that the the designed network of CPGs, is able to maintain a desired mean velocity of locomotion, with at least some restrictions. It still lacks the capability of successful adapting to a multitude of different scenarios. This comes mainly from the requirement of a careful analysis of the chosen state and control variables.

Finally, the initial conditions of the experiment could also change the final outcome, though as long as the leg maintains the ability to eventually achieve the desired overall velocity, which seems to be the case, that should not be a problem.

7.1 Future work

The experiment should be repeated with a change on the order of the lift mass and floor type alterations, as well as their timing. Also, the desired velocity chosen on this particular case was due to the fact that it was one the few possible when one looks at the overlap of the velocities achieved for each experiment of the first stage. In addition, the implementation of the control system in different types of locomotion system's should provide more insight in its viability. It should also be noted that the control approach is generalized — it can be used to control others aspects of locomotion. While its automatability can be hard to achieve, the control possibilities it provides is to great to be left unnoted. The ability to tune the defining parameters of a system's locomotion without turning to the use of tools that require lengthly and heavy optimizing processes is very appealing.

Finally, the combination of the proposed control architecture with other approaches, such as a feet placement locomotion gaits, that have to ability to guide a locomotion system in situations where the current perturbations are far to aggravating to rely on an optimization tool based on such a on the fly, low amount of collected training data.

Bibliography

[1] http://www.malab.se.ritsumei.ac.jp/en/robot/snake-like-robot/cpg-based-neural-controller-for-serpentine-locomotion/, last checked november 12, 2012.

[2] S. Arimoto, S. Kawamura, and F. Miyazaki. Bettering operation of dynamic systems by learning: a new control theory for servomechanism or mechatronics system. In *Proc. 23rd IEEE Conference on Decision and Control*, 1984.

[3] S. Arimoto, S. Kawamura, and F. Miyazaki. Learning control theory for dynamical systems. In *Proc. 24th IEEE Conference on Decision and Control*, 1985.

[4] Christopher G. Atkeson, Andrew W. Moore, and Stefan Schaal. Locally weighted learning. *Artificial Intelligence Review Artificial Intelligence Review Artificial Intelligence Review*, 11:11–73, 1997.

[5] Christopher G. Atkeson, Andrew W. Moore, and Stefan Schaal. Locally weighted learning for control. Technical report, College of Computing, Georgia Institute of Technology, 1999.

[6] Cynthia Breazeal. Toward social robots. *Robotics and Autonomous Systems*, 42:167--175, 2003.

[7] Samprit Chatterjee and Ali S. Hadi. Influential observations, high leverage points, and outliers in linear regression. *Statistical Science*, 1:379–416, 1986.

[8] David John Christensen, J. Christian Larsen, and Kasper Stoy. Adaptive strategy for online gait learning evaluated on the polymorphic robotic LocoKit. In *IEEE Conference on Evolving and Adaptive Intelligent Systems (EAIS)*, pages 63–68, 2012.

[9] P. Dyer and S. R. McReynolds. *The Computation and Theory of Optimal Control*. New York: Academic, 1970.

[10] Gen Endo, Jun Morimoto, Takamitsu Matsubara, Jun Nakanishi, and Gordon Cheng. Learning CPG-based biped locomotion with a policy gradient method: application to a humanoid robot. *The International Journal of Robotics Research*, 27(2):213–228, February 2008.

[11] Andrej Gams, Auke J. Ijspeert, Stefan Schaal, and Jadran Lenareice. On-line learning and modulation of periodic movements with nonlinear dynamical systems. *Auton Robot*, 27:3–23, 2009.

[12] Sten Grillner. Locomotion in vertebrates: central mechanisms and reflex interaction. *Physiological Reviews*, 55:247–304, 1975.

[13] Sten Grillner. *Handbook of Physiology, The Nervous System, 2, Motor Control*, chapter Control of locomotion in bipeds, tetrapods and fish, pages 1179–1236. American

Physiology Society, Bethesda, 1981.

[14] Sten Grillner, Peter Wallén, Kazuya Saitoha, Alexander Kozlov, and Brita Robertson. Neural bases of goal-directed locomotion in vertebrates – An overview. *Brain Research Reviews*, 57:2–12, 2008.

[15] M. Hildebrand. Analysis of asymmetrical gaits. *Journal of Mammology*, 58(2):131–156, May 1977.

[16] Auke Jan Ijspeert. Central pattern generators for locomotion control in animals and robots: A review. *Neural Networks*, 21:642–653, 2008.

[17] Auke Jan Ijspeert, Jun Nakanishi, and Stefan Schaal. Trajectory formation for imitation with nonlinear dynamical systems. In *IEEE International Conference on Intelligent Robot and Systems*, pages 752–757, 2001.

[18] Auke Jan Ijspeert, Jun Nakanishi, and Stefan Schaal. Movement imitation with nonlinear dynamical systems in humanoid robots. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA2002)*, pages 1398–1403, 2002.

[19] Eric R. Kandel, James H. Schwartz, and Thomas M. Jessel. *Principles of Neural Science*. McGraw-Hill, 2000.

[20] Sadao Kawamura and Norihisa Fukao. A time-scale interpolation for input torque patterns obtained through learning control on constrained robot motions. In *IEEE International Conference on Robotics and Automation*, volume 2, pages 2156–2161, 21-27 May 1995.

[21] O. Kiehn. Locomotor circuits in the mammalian spinal cord. *Annu Rev Neurosci*, 29:279–306, 2006.

[22] M. MacKay-Lyons. M. mackay-lyons. central pattern generation of locomotion: a review of the evidence. *Phys Ther*, 82(1):69–83, 2002.

[23] Takamitsu Matsubara, Jun Morimoto, Jun Nakanishi, Sang-Ho Hyon, Joshua G. Hale, and Gordon Cheng. Learning to acquire whole-body humanoid center of mass movements to achieve dynamic tasks. *Advanced Robotics*, 22:1125–1142, 2008.

[24] Olivier Michel. Webots: Professional mobile robot simulation. *International Journal of Advanced Robotic Systems*, 1(1):39–42, 2004.

[25] Takeshi Mori, Yutaka Nakamura, Masa aki Sato, and Shin Ishii. Reinforcement learning for a CPG-driven biped robot. In *AAAI'04 Proceedings of the 19th national conference on Artifical intelligence*, pages 623–630, 2004.

[26] Jun Nakanishi, Jay A. Farrell, and Stefan Schaal. Composite adaptive control with locally weighted statistical learning. *Neural Networks*, 18:71–90, 2005.

[27] Jun Nakanishi, Jun Morimoto, Gen Endo, Gordon Cheng, Stefan Schaal, and Mitsuo Kawato. Learning from demonstration and adaptation of biped locomotion. *Robotics and Autonomous Systems*, 47:79–91, 2004.

[28] G. Keir Pearson. Generating the walking gait: role of sensory feedback. *Progress in Brain Research*, 143:123–129, 2004.

[29] Ludovic Righetti and Auke Jan Ijspeert. Design methodologies for central pattern gen-

erators: an application to crawling humanoids. In *Proceedings of Robotics: Science and Systems*, pages 191–198, Philadelphia, USA, 2006.

[30] Ludovic Righetti and Auke Jan Ijspeert. Pattern generators with sensory feedback for the control of quadruped locomotion. In *IEEE International Conference on Robotics and Automatation*, 2008.

[31] S. Rossignol, R. Dubuc, and J. P. Gossard. Dynamic sensorimotor interactions in locomotion. *Physiol Rev.*, 86(1):89–154, 2006.

[32] Seyed Mojtaba Saif. Central pattern generator parameter search for a biped walking robot. *European Journal of Scientific Research*, 57(3):466–477, 2011.

[33] Takahiro Sato, Kota Watanabe, and Hajime Igarashi. Acquisition of adaptive walking behaviors using machine learning with central pattern generator. In *International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2010.

[34] Stefan Schaal, Peyman Mohajerian, and Auke Jan Ijspeert. Dynamics systems vs. optimal control - a unifying view. *Progress in Brain Research*, 165:425–445, 2007.

[35] Klaus Schittkowski. *Numerical data fitting in dynamical systems: a pratical introduction with applications and software*. Springer, 2002.

[36] Alexander Sproewitz, Rico Moeckel, Jérôme Maye, and Auke Jan Ijspeert. Learning ro move in modular robots using central pattern generators and online optimization. *The International Journal of Robotics Research*, 27(3–4):423–443, March 2008.

[37] Richard S. Sutton and Andrew G. Barto. *Reinforcement learning: an introduction*. MIT

Press, 1998.

[38] Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In MIT Press, editor, *Advances in Neural Information Processing Systems 12*, pages 1057–1063, 2000.

[39] Yakov Z. Tsypkin and S. J. Nikolic. *Adaptation and Learning in Automatic Systems*. Academic Press, Inc., 1971.

[40] B. Vanderborght. *Dynamic stabilisation of the biped Lucy powered by actuators with controllable stiffness.* PhD thesis, Vrije Universiteit Brussel, June 2007.

[41] Sethu Vijayakumar, A. D'Souza, and Stefan Schaal. ncremental online learning in high dimensions. Technical report, School of Informatics, University of Edinburgh, 2005.

[42] Sethu Vijayakumar and Stefan Schaal. Locally weighted projection regression : an o(n) algorithm for incremental real time learning in high dimensional space. *Proceedings of the Seventeenth International Conference on Machine Learning (ICML2000)*, pages 1079–1086, 2000.

[43] M. Vukobratovic and B. Borovac. Zero-moment point — thirty five years of its life. *International Journal of Humanoind Robots*, 1(1):157–173, 2004.

[44] Matthew M. Williamson. Neural control of rhythmic arm movements. *Neural Networks*, 11(7-8):1379–1394, 1998.